Assessing user specifications of robot behaviour for material transport tasks

by

Alexandru Blidaru

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Robots are an established component of many existing manufacturing processes. In the majority of cases, these robots operate in segregated areas, separated from human workers. However, as robots' capabilities in sensing and autonomy improve, they are expected to increasingly operate in human environments, and interact with novice, untrained users. When robots operate in human or shared environments, their tasks and behaviours need to be specified. This is typically performed by a human operator or supervisor, who may specify constraints on robot behaviour to make the robot more predictable or align its behaviour with user expectations. However, these constraints may impact robot task performance.

This thesis develops a user interface to obtain robot specifications, and proposes metrics for quantifying the specification quality, and to investigate how users create robot specifications. The metrics relate the robot's performance in the specified environment to its performance in a fully-unconstrained environment, and capture the trade-offs that users make in ensuring that the robot accomplishes its tasks while minimizing the loss of performance.

The proposed approach is evaluated in a series of user studies. The first user study sought to understand how novice users provide specifications for an autonomous robot operating in a shared warehouse environment, and to validate the metrics by applying them to user-created specifications. The metrics were then modified based on the results of the pilot study, and employed in a second, larger study. The second study trialed a modified interface and interaction scheme that implemented an interactive preference learning system, aimed at modifying specifications to improve robot performance. The modified metrics were then used to assess the quality of specifications following the preference learning system.

The two studies show that inexperienced users create a wide variety of behaviour-limiting specifications, and that they generally have difficulty creating efficient specifications, or assessing their own performance. Furthermore, the preference learning process succeeds in improving specification quality by making them more efficient and more similar between different users. Moreover, users that created specifications of worse initial quality benefit the most from the interactive learning process, as those specifications see a larger improvement.

Acknowledgements

I would like to first thank my supervisors, Dana Kulić and Stephen Smith, for providing me with countless opportunities, and terrific advice, feedback and guidance. They are exemplary researchers and supervisors, dedicated and passionate about their work, making them obvious sources of inspiration.

This thesis would not have been possible without the collaboration of Nils Wilde, and all of his effort in understanding how users specify robot behaviour. He has made an excellent research partner, who has had much to teach me. Despite all this, if we never had to deal with ROS installation issues, it would be too soon. Furthermore, I'd also like to thank Ryan Gariepy, CTO of Clearpath Robotics, who has helped guide our project. His insights into true user behaviour were invaluable.

I would like to thank everyone in the Adaptive Systems lab, as well as the Smith Autonomy Lab for your generous help, and inspiration. You have all taught me about so many different things throughout the years, some that I wouldn't even have dreamt of. I would like to offer special thanks to Pamela Carreño, Vladimir Joukov, Jonathan Lin, Terry Taewoong Um, Brandon J. DeHart, Alex Botros, Sagar Rajendran, for all your advice, help and support.

Thank you Jason Zeng, Ginette Tseng and Jonathan Lin for convincing me to embark on this journey.

I am grateful for my friends and family, who have supported me throughout my schooling, even when there was not much to celebrate. In particular, I would like to thank Tarryn LeBlanc, Raluca Blidaru, Narcis Blidaru, Teodora Blidaru, Domnica and Petre Mihalcea, Akshat Agarwal, Christian Teixeira, Nicholas Doan, Abhiram Bojadla.

Finally, thank you to everyone else who I haven't mentioned here, but who's helped me one way or another along the way. You have all impacted me and shaped my experience, and for that, thank you.

.

Dedication

This is dedicated to my friends and family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Conventional robotic technology has been designed with highly trained users in mind. As robots achieve higher levels of autonomy in a variety of environments, robot designers can no longer assume a high level of expertise from all potential users, and instead they must create human-robot interfaces that enable simple and effective communication between robots and untrained users.

Existing mobile autonomous robots are able to successfully plan and navigate in controlled environments [2]. Generally the robot planner is designed to find paths that minimize a certain cost function, usually the distance or time of travel. For a human working in the same environment as the robot, it might be difficult to easily understand and predict its behaviour. As a result, it is often preferred for the behaviour of the robot to be limited or structured in some manner. This reduces the complexity and number of actions that the robot can take at any given time, making its behaviour more predictable. Additionally, users might also expect the robot to follow existing conventions, such as road rules. These two factors (higher predictability and alignment of expectations) have been previously found to correlate to higher trust in robotic systems [3, 4], which in turn leads to increased usage [5, 4, 6] and increased effectiveness [7].

An easy way to constrain a robot's behaviour is to designate special-behaviour zones in the environment. The robot can then take these areas into consideration during planning, and follow the rules of the zone when inside it.

Before a robot can begin autonomous operation in a new environment, the desired zones need to be specified by human operators. This process however, can be challenging, especially for novice and untrained users. These users may not be familiar with the robot's capabilities, which can make it difficult for them to encode the expected robot behaviour.

In addition, due to their inexperience, it can also be more difficult for these users to appreciate the impact of their specifications on the performance of the robotic system.

A motivating example, and the focus of our work, comes from industrial and warehouse robotics. In such environments, mobile robotic platforms are used to transport material. For example, line-following automated guided vehicles (AGVs) are used in a variety of manufacturing plants. However, AGV trajectories are rigidly defined by magnetic tape lines on the ground, so it is hard and expensive to reconfigure a system once installed. Fully autonomous robotic platforms that are not limited to specific trajectories are a promising alternative. While AGVs only follow specific laid out paths, non-constrained mobile autonomous platforms can move anywhere in the open space and generate their paths autonomously. To ensure safety and predictability, additional specifications are required to ensure that the robots follow existing conventions, customs and rules. Examples include driving on the correct side of the road, stopping before intersections, avoiding certain areas, and following established directions of travel down narrow hallways and aisles. Operators tasked with managing the autonomous robot fleet should have the capability to easily create specifications that restrict the behaviour of the robot to enforce the desired processes and rules.

In this thesis, we develop a set of metrics that allow us to evaluate and quantify the quality of user specifications. The proposed metrics are validated by applying them to several sets of user-created specifications, which were obtained through a user study where participants took the role of a mobile robot fleet manager and interacted with a simulated robotic system. Given the user specification on the environment map, a graph representing the environment and user constraints is obtained by encoding the constraints into a state lattice. The proposed metrics are applied to the generated motion graph. The metrics capture the positive (i.e. increased predictability of robot behaviour) and negative (i.e. non-optimal behaviour compared to an environment devoid of any restrictions) effects of a user specification. These metrics allow us to capture the trade-offs that users make in ensuring that the robot accomplishes its tasks while minimizing loss of performance. The ability to assess user specifications could then be used to help users improve their specifications.

The main contribution of this thesis lies in the set of metrics to assess user-specified robot behaviour (originally published in [8]), which were validated through two user studies (published in [8, 9]). The metrics provide a way to quantitatively assess a user specification, capturing the trade-offs between the benefits and costs of constraining robot behaviour. Unlike existing work, the proposed metrics enable a characterization of both the global robot performance in the environment, and the task-specific performance defined on a set of start-goal pairs. These metrics are then used as part of two user studies. Through the initial

user study we demonstrate that users unknowingly create specifications that achieve a wide range of robot performance. This finding underlines the need for an interaction scheme that empowers users of various expertise levels to create efficient specifications, unlike many existing industrial robotics applications where users require specialized training. In our second study, we employ an active preference learning system that is designed to improve the quality of user specifications, and validate it using our metrics. Our results demonstrate that the preference learning algorithm can significantly improve specification performance with few iterations of user interaction, especially in the case of specifications that were most detrimental to robot performance. The second user study was performed in conjunction with Nils Wilde, a PhD candidate at the University of Waterloo. In this joint work, Nils proposed the learning algorithm, and was the lead on this aspect of data analysis. I led the development of the user study and interface, the application of the metrics to the data and associated data analysis.

1.1 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 provides an overview of the related work, concerning user interaction modalities, performance metrics, and task specifications.

In Chapter 3 we additionally provide background information on the functionality and operation of the interactive preference learning system employed in the user study.

Chapter 4 describes the development of the interface through which an operator can specify robot behaviour. As part of this discussion, we first present the functional requirements of our user interface, followed by an identification of design elements commonly found in performant interfaces. Finally, we present our interface implementation, discussing its capabilities.

Chapter 5 describes the initial development of metrics to quantify a specification's performance. In this chapter, we define our chosen metrics, and then apply them on user-authored specifications that were obtained through a preliminary user study.

Building on the pilot results obtained in the previous chapter, Chapter 6 describes a large scale user study integrating specification learning. In this study, we integrate a preference learning system that allows us to learn the importance that participants place on the constraints that make up their specification. The robot can then violate these the less important constraints when it is sufficiently beneficial, resulting in improved performance as captured through our metrics.

In Chapter 7, we present the overall conclusions of our two studies. Additionally we suggest several potential future directions for the research. These include more representative user studies, and an improved specification process, allowing us to more easily improve the performance gains possible with the overall system.

Chapter 2

Related Work

As part of this thesis, we design and implement 1) a user interface that allows users to specify robot behaviour, and to interactively provide feedback improve robot performance, and 2) a set of metrics to quantify the performance of the robot given the user specification. As such, the work presented in this thesis encompasses several areas of robotics, whose existing work can guide our own efforts: human-robot interaction, performance assessment, and task specification. In this chapter, we first review existing work in human-robot interaction, to guide the design of the appropriate interaction system for our application. Next, existing performance metrics relevant for assessing user specifications are reviewed. Then, related work in task specification is identified, which can guide our efforts to obtain behaviour specifications, and improve existing behaviour specifications from users.

2.1 User Interaction Modalities

Multiple works have sought to better understand the requirements of effective HRI, by creating taxonomies related to the type of robots, types of users, or types of interactions. Understanding where a specific robotic application fits into the proposed taxonomies can be used to identify the key concern and elements leading to a successful interaction scheme. This is especially important as robots begin to interact with a wider subset of the population, the majority of which will not be robotics experts. In this section we review existing work aimed at categorizing user interaction based on shared attributes.

In [10], Thrun suggests that a robot's level of autonomy is a key consideration for effective HRI, as it determines the tasks that the robotic system can perform, as well as the

level at which the interaction takes place. In exemplifying the importance of autonomy, the authors consider three broad groups of robots: industrial robots (low autonomy level), professional service, and personal service robots (higher autonomy levels). The interfaces and robot interactions vary depending on the autonomy level, with industrial robots needing interfaces that consider their limited autonomy, and separated workspace, while most service robots need richer interfaces that need to support direct (i.e. robot acting on its own), and indirect (i.e. robot executes operator commands) autonomy. Thrun proceeds to illustrate several open questions at the time of writing, of which two are directly applicable to our work: "What happens when robots work with or interact with groups of people?...", and "To what extent is progress in human-robot interaction tied to progress in other core disciplines of robotics, such as autonomy? In which way will future advances in robotic autonomy change the way we interact with robots?". Regarding the first question, our robotic system is envisioned to work and interact with different groups of users, for example, operators (of various skill levels) setting the system up, as well as other human co-workers on the plant floor. As such, interaction with the robot needs to accommodate and be performant with respect to all of these users. Regarding the level of autonomy, our robotic system is capable of autonomous navigation, operating in a shared workspace, and interacting with people with little knowledge of robotics, therefore fitting into the service robot level.

In [11], Scholtz creates a taxonomy defining five models of human-robot interaction: supervisor (monitoring and controlling the overall situation), operator (modifying internal models to correct robot behaviour), mechanic (managing physical interventions), peer (giving commands within larger goal/intentions context), and bystander (limited interaction with the robot). A human operating in a joint human-robot system will have different tasks, and situational awareness needs depending on their interaction role. Additionally, Scholtz highlights that the boundaries between the layers of interaction are fuzzy, and that humans could be tasked with interacting with the robot through more than just one of the interaction models. This is true in our application as well, where individuals are expected to interact with the robot in different roles concurrently. For example, the robot-warehouse integrator tasked with setting up the system and supervising its operation will be interacting with the robot in the supervisor and operator roles, as they will be tasked with creating and modifying the constraints on robot behaviour. On the other hand, a shift manager's interaction with the robot will be mostly covered by the peer model, with some operator tasks as needed if the robot diverges from expected behaviour and intervention is needed. Depending on the current robot activity, floor workers would interact with the robots under as peers or bystanders; as a peer, the worker could correct robot behaviour in specific areas (e.g. worker telling a robot not to use a certain aisle due to obstacles beyond

the sensing range of the robot), while as a bystander, the worker will need to possess a minimum understanding of robot behaviour since they are both co-existing in the same workspace.

In their survey of HRI [12], in addition to considering the user roles defined in [11], Goodrich et al. further separate interaction into two general categories: remote interactions when the human and robot are separated spatially/temporally, and proximate interactions when the human and robot are co-located. Furthermore, they identify five attributes that robotics designers can vary to affect HRI:

- level and behaviour of autonomy,
- nature of information exchange,
- structure of the team,
- adaptation, learning, and training of people and the robot,
- shape of the task.

These five dimensions are then summarized by the concept of dynamic interaction, which includes time- and task-varying changes to the above. Our work is primarily focused on the second and fourth attributes of dynamic interaction, as we are trying to better understand the requirements leading to a robotic system that can easily and effectively learn users' preferences with regards to its behaviour.

Based on these taxonomies, we can identify several elements that are important for designing human-robot interaction systems, including the type of users and the tasks and capabilities that are best performed by each user type, and the level of autonomy possessed by the robotic system. Additionally the interaction with the robot system is also affected by the level of adaptation, and learning between users and the robot.

2.2 Performance Metrics

In this section we present existing work dealing with the measurement of performance in robotic systems. Due to the large variety of robotics use-cases and component technologies, it is difficult to identify fully generalizable metrics. One solution to this problem involves the creation of standardized tasks and scenarios, something that is most commonly seen with urban search and rescue robots [13]. Alternatively, the HRI problem can be broken

into sub-components, and applicable metrics can be found across each component. We will first discuss metrics built on top of taxonomies, followed by taxonomy-independent metrics based on the separate robot, human, and overall system performance of a robotic system.

For example, Steinfeld et al. [13], identify five categories of robot tasks for which common metrics were proposed: navigation, perception, management, manipulation, and social. For the navigation task, the authors propose measuring the effectiveness (e.g. percentage of tasks completed, coverage area, deviations, avoided obstacles, etc.), and efficiency (e.g. time to task completion, operator time, etc.) related to completing the task. In terms of management, Steinfeld et al. suggest measuring the number of robots that could be effectively controlled by a human, as well as identifying failures due to autonomy discrepancies. Finally, regarding the social aspect, it is suggested to keep track of the interaction characteristics (e.g style, social context), as well as the level of persuasiveness, trust, engagement, and compliance of the robot.

Alternatively, to make metrics taxonomy-independent, HRI performance can be investigated based on the following three aspects: system, human, and robot.

System performance primarily deals with the effectiveness of the robot-human team. One common way to measure it is to determine the task-specific performance (i.e. does the system succeed in accomplishing its task) [13, 14]. However, Steinfeld et al. argue that task-specific performance can be misleading in certain situations, as it does not take in consideration the autonomy of the system [13]. For example, a scenario can be envisioned where a robotic system is designed to be fully autonomous, though in reality it fails a certain percent of the time, requiring the human operator to pick up the slack. Looking at just the overall task success rate would then hide the autonomy failure for which the human compensated, as there would be no way of knowing if a success was due to perfect autonomous operation (i.e. a true success of the full robot autonomy), or due to the human intervention (i.e. a failure of what should be full automation). In addition to quantitatively measuring performance, subjective ratings can also be used to assess the system performance [13].

Several aspects of operator performance exist, and although many of them were initially developed in the context of aircraft operations [14], they have since been applied widely to many situation dealing with human performance. Amongst these, situational awareness (SA) -generally measured through the Situtation Awareness Global Assessment Technique (SAGAT) [15]- has been found to be of critical importance to operator performance in robotics [16, 17]. In addition, SA is also affected by the autonomy and interaction with the robot [18], as increased robot autonomy makes it easier for a human to lose situational awareness, linking operator performance with robot performance. Workload -often mea-

sured through the NASA-Task Load Index (NASA-TLX) [19]- is another factor that affects human performance [20, 13], with higher workload leading to a decrease in a user's SA. The accuracy of mental models of device operation, such as design affordances, or operator expectations can also impact human performance [13]. By developing interactions and interfaces that mimic the mental models that humans have of a robot, they avoid having to perform any mental transformations reducing the cognitive load [13, 21, 22].

With regards to the robot's performance during HRI, Steinfeld et al. propose measuring a robot's self-awareness (i.e. knowing when to involve a human in its operation), human awareness (i.e. being aware of humans and/or their commands, expectations, constraints, and intent), and autonomy (i.e. a robot's ability to function on their own) [13]. A robot's performance can also be ascertained by measuring the complexity of it's operating environment [23, 24]. In [25, 26, 27, 28] it was suggested that complexity be determined by approximating the branching factor and amount of clutter in the environment. The work of [24, 29, 30] proposes a technique rooted in information theory to determine the robot operating environment, with [24] measuring entropy based on obstacle density, while [29, 30] used the number of accessible neighbours at every location in the motion graph. Additionally, [30] proposes a secondary complexity measure based on the distribution of obstacle types and the compressability of the environment. In [31], the measurement of complexity was extended from a binary distribution of local obstacles to a continuous one, enabling the use of dynamic obstacles.

As there are no standardized tasks or scenarios for material transport tasks, the focus of our work will be on generating component-based metrics. As part of this, we consider both taxonomy-dependent approaches (as our task is ultimately one of navigation), as well as taxonomy-independent approaches where we can look at robot- human- and joint-performance separately.

2.3 Task Specification

In this section[1] we present existing work related to task specification. We first discuss related work on obtaining specifications from experts, followed by related work on specification revision. Finally, we examine existing research on active preference learning. This section is pertinent to the active preference learning system employed in Chapter 6, where we attempt to revise user specifications to improve robot performance.

[1] Adapted from [9]

2.3.1 Specifications Obtained From Experts

First we review methods for task specification where an expert operator specifies a robot task by either defining reward functions, providing optimal demonstrations or using a specification language.

In the first method, reward functions are used to describe the high-level behaviour for the robot, which then learns the appropriate policy using reinforcement learning (RL) [32, 33]. A user defined reward function maps the system states to a numerical value, expressing how desired that state is. This reward function corresponds to a high-level specification for how the robot should behave; through RL the robot then finds a policy that maximizes the reward. RL has been extensively studied as a tool to realize a high level description of a robot's behaviour [32]. For instance, [33] and [34] apply RL in the domain of mobile robots and robots competing in soccer games. In both examples the reward function is designed by a human expert. The practicality of RL approaches has also been investigated in field studies [35]. However, specifying reward functions usually requires a high level of expertise and can be unintuitive.

The field of learning from demonstration (LfD) uses expert demonstrations for robot programming [36, 37]. Applications range from high-level task specification [38] to the definition of precise actions such as grasping [39] or manipulation trajectories [40]. A common technique in LfD systems is inverse reinforcement learning (IRL) [41]. The setting is similar to RL; however, the reward function is unknown. When using IRL in LfD, the objective is to learn how the robot should behave. Demonstrations are provided by a human expert; it is assumed that the human maximizes an internal reward function [42, 41]. From multiple demonstrations the learning system tries to recover that reward function in order to imitate the behaviour. The reward function is often modelled as a linear combination of pre-defined features; the problem then consists of learning the weights for all features [42]. However, in practice LfD faces challenges when demonstrating the desired behaviour requires a high level of expertise [43] or a large number of examples [44].

Specification languages such as linear temporal logic (LTL) [45] allow for abstract specifications, for instance *"First, visit region A and B, then go to C and finally visit D"*. In order to reduce the burden on the user, [46] proposes a GUI for LTL mission planning, while [47] designs a framework for using natural language to provide LTL specifications.

This relates to the IRL problem at the core of the work in [9] (used in Chapter 6) where we want to learn a user's cost function for the constraints they specified, i.e., their importance, thereby improving robot performance. Any path that is generated between the start and goal location could be described by a set of features, including ones that

describe the violation of constraints. Then, learning about the importance of constraints is analogous to recovering a user reward function based on these features.

2.3.2 Revision Of Specifications

The second approach takes into consideration that demonstrations and specifications – especially when provided prior to the robot executing the task – might be sub-optimal. In the field of LTL, the works of [48] and [49] both revise an initial specification if it leads to sub-optimal outcomes or is infeasible. In a general motion planning problem on some configuration space with spatial obstacles, [50] considers the case when no feasible path exists. The minimal constraint removal problem then finds the biggest subset of obstacles such that a feasibility is re-attained.

The concept of revising initial specifications is also applied to LfD. The work of [51] automatically segments the tasks and then efficiently asks for additional demonstrations when needed. Moreover, [52] focuses on failed demonstrations. Instead of imitating the human, the learning system tries to avoid repeating the mistakes the operator made.

In a comparable fashion, in the procedure presented in Chapter 6, we receive a set of constraints from a user. Then, using the learning approach developed by Wilde et al. in [53], we initially set high weights for all constraints such that the resulting path respects all user constraints to yield an initial specification. However, we assume that such a path might not necessarily be optimal, as some constraint violations might be allowable, shortening the path length. The user agreeing to the violation of a constraint can be thought of as relaxing the constraint in question, leading to a revised specification. Due to the constraint relaxation, this revised specification is hypothesized to exhibit improved robot-performance, something that could be shown by our metrics.

2.3.3 Active Preference Learning

More recently, research has focused on defining the desired behaviour of a robot interactively [54, 55, 56, 57, 44, 58]. In active preference learning, users are presented with possible solutions for a defined problem. When they choose between alternatives, the autonomous system learns about their preferences and iteratively improves its strategy. Interactive task specification addresses several drawbacks of the previously discussed techniques. For instance, asking a user for demonstrations is not always desirable, as human

demonstrations can be infeasible, e.g. in swarm robotics [59], difficult to provide [54, 41], the amount of necessary demonstrations may be prohibitively large [44] or the demonstration itself requires a high level of expertise [54]. Providing rich and precise specifications prior to a robot executing a task might also be challenging and more prone to inaccuracies [41]. Interactive task specification also improves ease of use by reducing the information required from the user up front. Instead of asking the user for a complete specification in the beginning or demanding numerous demonstrations, robot tasks can be learned in an iterative, interactive way.
The work of [57, 44] addresses these challenges by integrating user feedback into RL systems. Also focusing on RL for autonomous robots, [60] investigate how different interactive learning algorithms are accepted by users and show that users perceive action advice as more effective than action critique in a study with 24 participants. [44] apply user interaction to RL. Instead of using human feedback as a reward function, users are asked for their pair-wise preference for possible trajectories. This allows to drastically reduce the amount of necessary user interaction.

Recently, numerous contributions to interactive task specification have been made in the field of active preference learning, combining techniques from preference elicitation [61, 62] and active learning [63, 55]. The problem of preference elicitation considers a set of hypotheses, tests and outcomes. By performing tests, some hypotheses become inconsistent with the observed outcomes and are rejected. This can be applied to a robot task specification: Hypotheses are possible reward functions of the user. Tests correspond to presenting the user with alternative solutions based on these reward functions, while observations are the user's selections. The user's internal reward function is then learned by iteratively ruling out reward functions that become inconsistent with the user's choices.

Active learning allows the learner to decide what query, i.e., what set of alternative solutions the user is presented with next. [54] present a framework where experts rank the performance of a demonstrated grasping task.

In [55] and [64], trajectories for a dynamical system are presented to the user, who then chooses one of two alternatives. Iteratively, weights for trajectory features are learned and an optimal solution is found. [55] validate their work in simulation and in a small user study (10 participants). In both experiments the user model is based on 5 predefined features. While the simulations demonstrated the convergence of their algorithm over 200 iterations, the user study showed subjective improvements over 10 iterations. The subsequent work of [64] with richer user feedback is supported by another study with 10 participants that interacted with the learning learning system for 20 iterations.

The framework of Wilde et al. used in Chapter 6 is based on active preference learning. Through it, we query the user about their preference for alternative paths and learn about the importance of user constraints from their feedback. However, in our case the set of hypotheses is the set of all possible paths between the start and goal, which is not directly given. When planning on a graph, finding the set of all paths from start to goal is known to be a #P-complete problem [65]. Other work in the field of active preference learning often focuses on user preferences about the robot's behaviour itself, e.g., [55]. In contrast, we consider user preferences about the environment the robot acts in. Moreover, in the scenario presented in Chapter 6 we have explicit prior information about the user's preferences. We assume they follow two objectives: minimizing time and only allowing constraint violation when sufficiently beneficial. This may allow us to design strategies for presenting the alternative paths that are either greedily maximizing the potential information gain or that are likely to be accepted by the user, resulting in significant improvements in robot performance.

2.4 Summary

This chapter provided an overview of related work on user interaction modalities, performance metrics, and various forms of task specification. We identified multiple taxonomies used in classifying user interaction, classifications that can, in turn, help us understand the requirements of good interaction schemes. These taxonomies can then be used to design metrics that are applicable across each category of a taxonomy. Alternatively, taxonomy-independent metrics can be created, where the robot, human, and system performance is assessed separately. We presented a variety of ways to specify tasks to a robotic system, including that of active preference learning which is the category that the approach used in Chapter 6 falls in.

Chapter 3

Background Work

Part of the work presented in this thesis uses the work of Wilde et al. in [53], which develops an approach for learning user preferences for complex task specifications through human-robot interaction. As such, in this chapter a more detailed overview of the work is provided to aide in fully understanding this thesis.

The work of Wilde et al. considers the problem of navigating through a known environment, where a user has specified the robot task by applying a set of constraints on allowable robot behaviour. These constraints however can greatly impact the performance of the robotic system, and may not all be considered equally important by the user. As such, a user-on-the-loop method to learn the user's preference regarding these constraints is developed. The algorithm works by generating an initial set of solutions, and then iteratively generating alternative paths, which are then presented to the user, who is asked to provide feedback on these paths. It is assumed that users have a hidden cost for each constraint, which expresses the time benefit for which a violation will be accepted. By iteratively prompting the user with alternative paths and requesting their feedback, the algorithm learns the range that the hidden costs lie in, ultimately approaching a unique solution for the shortest path problem, and thus sufficiently learning the user preferences.

The problem has the following inputs:

- A single weighted strongly connected multi-graph $G' = (V, E, \Psi, t)$ which represents a robot's motion in an environment, where V is the set of vertices, E is the set of edges, the function Ψ associates each edge with an ordered pair of vertices, and the function t describes the traversal time for all edges. Parallel edges in the graph express different traversal speeds (i.e. different traversal times).

- A user specification consisting of n user constraints $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_n\}$. Each constraint is defined as the pair (E_i, w_i), with $E_i \subseteq E$, and w_i is a constant in $\mathbb{R}_{\geq 0}$ defined for all edges in E_i, which represents the weight of user preference in units of time.

- A start and goal vertex, v_{start} and v_{goal}, each in V.

When the hidden weights w_i are known, G' and Γ can be combined into a doubly weighted multigraph $G = (V, E, \Psi, t, w)$, where w is defined by:

$$w(e) = \sum_{\gamma_i \in \Gamma | e \in E_i} w_i \tag{3.1}$$

To simplify the graph above, all parallel edges where $w(e) = 0$ are deleted, with the exception of the one with minimal traversal time t. The goal is then to find an optimal path on G, from v_{start} to v_{goal}, to minimize t and w:

$$\min_p \sum_{e \in P} w(e) + t(e) \tag{3.2}$$

where w captures the increase in completion time that the user is willing to allow to satisfy the constraint. By picking an estimate for all w_i, the graph G collapses to a singly-weighted directed graph, where the shortest path can be efficiently computed. The effective goal of the approach is to find estimates for all constraint weights w, such that the corresponding path is optimal based on objective 3.2.

Using the above problem setup, Wilde et al. illustratively outline the procedure for learning user preferences in Figure 3.1. Given the problem inputs described above, a path $P(\hat{w}^0)$ from v_{start} to v_{end}, and that does not violate any user constraints is assumed to exist and is found. In each iteration, the path that received the best user feedback in the previous iteration (denoted by P^{best}, and initially set to $P(\hat{w}^0)$ is selected and executed. The system then generates alternative paths based on the learned user-preference weights w, and presents these alternative paths alongside P^{best} to the user for feedback. Once the user provides the requested feedback, the system updates its knowledge of the user-preference weights. The iterative process continues until either w is sufficiently learnt, or the system reached the maximum number of user interactions J.

The algorithm based on Figure 3.1 is presented as Algorithm 1. In line 1, the feasible space of weights (defined as a polyhedron of the form $\mathbf{A} \leq b$) is initialised with a finite upper bound w^{max}, which is also used to initialize w^{best} and to find the corresponding path

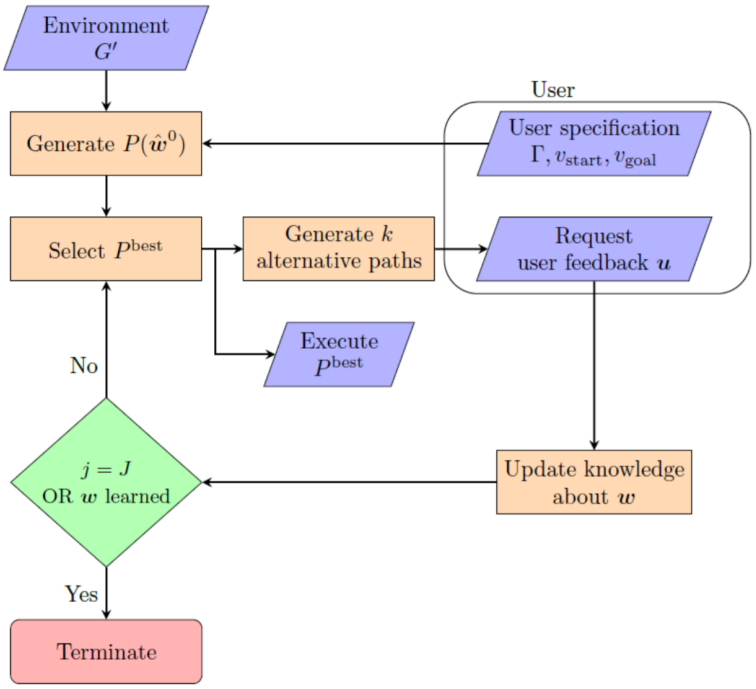


Figure 3.1: Flowchart of the learning process. j denotes the single iteration, J is the maximum number of user interactions. [1]

Input: G', Γ, J, k
Output: $\hat{\boldsymbol{w}}^{best}$

1 Initialize $\boldsymbol{A}$, $\boldsymbol{b}$, $\hat{\boldsymbol{w}}^{best}$, P^{best}, and $\mathcal{W} = \emptyset$
2 **for** $j = 1$ *to* J **do**
3 $\mathcal{W}_{new} \leftarrow \pi(\boldsymbol{A}, \boldsymbol{b}, k, \mathcal{W}, \boldsymbol{w}^{best})$
4 **if** $\mathcal{W}_{new} = \emptyset$ **then**
5 **return** $\hat{\boldsymbol{w}}^{best}$
6 **end**
7 Compute paths P^1,...,P^k for all $\hat{\boldsymbol{w}} \in \mathcal{W}_{new}$
8 Get user feedback $\boldsymbol{u}^j$ for paths P^{best}, P^1,...,P^k
9 Update $\boldsymbol{A}$ and $\boldsymbol{b}$ based on $\boldsymbol{u}^j$
10 Choose $\hat{\boldsymbol{w}}^{best}$ as the element from $\hat{\boldsymbol{w}}^{best} \cup \mathcal{W}_{new}$ with the best user feedback, $P^{best} = P(\hat{\boldsymbol{w}}^{best})$
11 $\mathcal{W} = \mathcal{W} \cup \mathcal{W}_{new}$
12 **end**
13 **return** $\hat{\boldsymbol{w}}^{best}$

Algorithm 1: Algorithm to learn new user weights[1]

P^{best}. In lines 3-5, k new weights are found according some admissible policy π that maps from $(\mathbf{A}, b, k, W, w^{best})$ to a set of k weights W^{new}. The policy is admissible if each of the new weights lay in the feasible weight space, and they are not equivalent to any previous weights in W, or any other weights in W_{new}. In lines 6-7, the alternative paths for each weight in W_{new} are computed, and then presented to the user alongside P^{best}. The user feedback is then used to update the feasible space (line 8), and the path with the best user feedback (line 9). Finally, all of the new weights are added to the set W.

Wilde et al. propose two policies in their work. In the first policy, dubbed $\pi_{vertexSearch}$, and presented in Algorithm 2, they use depth first search (DFS) to search the weight feasible space, starting at the previously best solution $\hat{w}^{best}$. The feasible space can be represented as an unknown graph, whose vertices are the extreme points of a polyhedron, and are connected together through its edges. In line 5 and lines 8-12, the neighbouring vertices are explored, a step that is computationally inexpensive and is similar to the pivot step in the simplex algorithm for a linear program. In lines 6-7, if a new vertex is found, it is added to W_{new}. The $\pi_{vertexSearch}$ algorithm stops when either k new vertices have been found, or when the DFS has either exhausted all vertices or reached a maximum number of iterations. However, this policy has the drawback of potentially exhausting an exponentially growing graph, leading to the need of a heuristic policy that would take into

account information from the shortest path problem. This policy, named $\pi_{minSearch}$, and illustrated in Algorithm 3, always attempts to minimize the weights in the current feasible space. It does so by first finding the minimal feasible weight, and adding it to W_{new} if the weight is not equivalent to any previous weights (lines 2-4). If the weight hasn't been previously found and $k = 1$, the algorithm terminates. Otherwise, the algorithm proceeds by inverting the search direction of all constraints violated by the path obtained from the minimal weight (line 7-11), and then uses the new search directions to generate new weights (line 13-15). If after all this, the heuristic has been proven to be unsuccessful, we revert to using the initial $\pi_{vertexSearch}$ policy.

Input: $\boldsymbol{A}$, $\boldsymbol{b}$,k, $\mathcal{W}$, $\hat{\boldsymbol{w}}^{best}$
Output: $\mathcal{W}_{new}$
1 Initialize set $\mathcal{W}_{new} = \emptyset$, `openList` $= \{\hat{\boldsymbol{w}}^{best}\}$ and maximum iterations i_{max}
2 **for** $i = 0$ to i_{max} **do**
3 **if** $|\mathcal{W}_{new}| = k$ or `openList` is *empty* **then**
4 **return** $\mathcal{W}_{new}$
5 **end**
6 $\tilde{\boldsymbol{w}}$ = `openList`.pop()
7 **if** $\tilde{\boldsymbol{w}}$ is not equivalent to any $\hat{\boldsymbol{w}} \in \mathcal{W}_{new} \cup \mathcal{W}$ **then**
8 Add $\tilde{\boldsymbol{w}}$ to $\mathcal{W}_{new}$
9 **end**
10 **if** $\tilde{\boldsymbol{w}}$ is not labelled as discovered **then**
11 Label $\tilde{\boldsymbol{w}}$ as discovered
12 **end**
13 **for** *all* $\boldsymbol{w}' \in$ getAdjacentVertices($\tilde{\boldsymbol{w}}$) **do**
14 **if** $\mathtt{w}' \notin$ `openList` and $\tilde{\boldsymbol{w}}$ is not labelled as discovered **then**
15 `openList`.insert($\boldsymbol{w}'$)
16 **end**
17 **end**
18 **end**
19 **return** $\mathcal{W}_{new}$

Algorithm 2: Algorithm for policy that finds new weights using DFS [1]

This leads to an algorithm that allows for the learning of user preferences in the context of robot navigation through a known environment, augmented with a set of known user-defined constraints of hidden cost.

To enable the specification process, and the subsequent preference learning process, we

Input: $\boldsymbol{A}$, $\boldsymbol{b}$,k, $\mathcal{W}$, $\hat{\boldsymbol{w}}^{best}$
Output: $\mathcal{W}_{new}$
$\mathcal{W}_{new} = \emptyset$
$\hat{\boldsymbol{w}} = \min \mathbf{1}^T \boldsymbol{w}$ s.t. $\boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{b}$
if $\hat{\boldsymbol{w}} \notin \mathcal{W}$ **then**
    add $\hat{\boldsymbol{w}}$ to $\mathcal{W}_{new}$
end
if $|\mathcal{W}_{new}| = k$ **then**
    return $\mathcal{W}_{new}$
end
`newSearchDirections` $= \emptyset$
for γ_i where $\phi_i(P(\hat{\boldsymbol{w}})) > 0$ **do**
    $\bar{\boldsymbol{c}} = \boldsymbol{c}$
    $\bar{\boldsymbol{c}}_i = -1$
    add $\bar{\boldsymbol{c}}$ to `newSearchDirections`
end
for $\bar{\boldsymbol{c}} \in$ `newSearchDirections` **do**
    $\tilde{w} = \min \bar{\boldsymbol{c}}^T \boldsymbol{w}$ s.t. $\boldsymbol{A}\boldsymbol{w} \leq \boldsymbol{b}$
    if $\tilde{\boldsymbol{w}} \notin \mathcal{W}$ **then**
        add $\tilde{\boldsymbol{w}}$ to $\mathcal{W}_{new}$
    end
    if $|\mathcal{W}_{new}| = k$ **then**
        return $\mathcal{W}_{new}$
    end
end
$\mathcal{W}' \leftarrow \pi_{vertexSearch}(\boldsymbol{A}, \boldsymbol{b}, k - |\mathcal{W}_{new}|, \mathcal{W} \cup \mathcal{W}_{new}, \hat{\boldsymbol{w}}$
return $\mathcal{W}_{new} \cup \mathcal{W}'$

Algorithm 3: Algorithm for policy minimizing new weights[1]

first have to design an interface which implements these functionalities. That process is the subject of the next chapter.

Chapter 4

Interface Design

The first objective of this research was the development of an application that could be used to capture a user's set of constraints on robotic behaviour. To create an effective interface, we first characterized the type of constraints to be captured, and the type of interaction that would best serve the creation of these constraints. Next, we analyzed findings from related work to extract design guidelines. Finally, the interface was implemented and validated.

4.1 Interface Specification Derivation

In this section, we derive the interface specifications that were used to design the user specification interface. The objective of the interface was to allow users to specify constraints on robot behaviour. Since the focus was limited to industrial robotics, specifically warehouse transport robotics, the majority of constraints were deemed to be area-based and planar, as industrial robots operate primarily on flat ground. Based on consultations with Clearpath Robotics, our industry partner, we determined that examples of these constraints could include directional constraints, speed constraints, or traffic preference constraints. We further assume that the environment in which the robot is operating is known ahead of time. During our consultation we learned that while some areas of the environment might see some small changes, the overall structure remains static, as this consistency helps with the retrieval and transportation of items. We also assume that the easiest way to specify constraints is by graphically marking them on the provided environmental map, an approach that has been previously implemented by systems focused on aerial robotics, where users are interested in designating "No Fly", or "Loitering" areas.

An early hypothesis regarding user specifications was that users will generally not be capable of creating efficient specifications, and as such, a method to improve specifications would be required. Towards that end, Nils Wilde, a PhD candidate, was focused on developing an interactive preference learning system that seeks to infer the importance that users place on their constraints. To avoid increasing user mental workload by having them interact with several applications, we decided to implement the preference learning system alongside the specification process such that users would interact with a single application. For the same reason, we decided that all study questionnaires to be conducted as part of the same interface.

We therefore derived the following functional requirements for our interface:

1. Floormap Visualization: Receive map file (i.e. occupancy grid file) and display a top-down view of the environment.
2. Specification Phase: Ability to create an initial specification that allows the user to create planar area-based constraints.
3. Learning Phase: Ability to perform interactive preference learning, allowing us to revise specifications, thereby improving robot performance.
4. Questionnaire Phase: Ability to run within and post-study questionnaires gathering performance-assessment and usability data.

4.2 Interface Guidelines

Although every robot needs an interface, and graphical user interfaces(UIs) have been widely used in interacting with robots, few papers discuss the choices behind the interface designs, and even fewer still investigate their usability. Determining what makes a good interface is further complicated by that fact that UIs are related to the functionality and goals that the robot is trying to achieve, and few areas of robotics have standardized scenarios and tasks.

One research area with standardized scenarios and tasks is Urban Search and Rescue (USAR), for which several robotics competitions were created (e.g. AAAI/RoboCup Robot Rescue Competition, DARPA Robotics Challenge). These competitions allowed for multiple independently-engineered robotic systems to be graded against the same set of tasks. This permitted researchers to perform comparative analysis between the user interfaces

employed by the competing teams, leading them to identify several guidelines in creating effective UIs. Based on the systems observed at the RoboCup Robot Rescue Competition, Yanco et al. identified the following guidelines to creating effective interactions/interfaces [66]:

1. Utilize a single monitor
2. Avoid small video windows
3. Avoid window occlusion
4. Use one robot to view another when possible
5. Design for the intended user, not the developer

At the DARPA DRC trials, Yanco et al. identified additional guidelines for effective interfaces/interaction [67]:

1. Increase sensor fusion
2. Decrease the number of operators
3. Decrease the amount of operator input needed to control the robot
4. Don't separate the robot into legs and arms
5. Plan for low bandwidth
6. Design for intended users

Following the DARPA DRC Finals, additional guidelines were created by Norton et al. based on the experiences of the teams observed [68]:

1. Balance operator and system capabilities to effectively perform the task
2. Keep operator in the loop
3. Maintain operator awareness of robot state, and use reliable and consistent control methods
4. Duplicate sensor fusion displays using different perspectives

5. Allow time for operator training and specialization

While these lessons were obtained from the analysis of teleoperated robotics systems focused on USAR, most of them can be interpreted as a series of best-practices and applied to robotics interfaces in other domains.

Another source of guidelines for our interface comes from various commercial aerial, and mobile industrial robotic platforms [69, 70, 71, 72]. Commercial offerings from aerial robotics (drone) companies generally provide the user a map-centered view of the robot and its systems. In their most simple forms, these interfaces display a map of the robot's surroundings, on top of which is overlayed the position of the robot, and information about its tasks. The operator interacts with the robot by marking areas of the map with the desired task or behaviour for the robots. For example, in the construction industry, an operator might designate areas on the map to be surveyed and mapped, letting the robot decide how to efficiently do that, or they could indicate specific goal locations and orientations for the robot to fly to and survey. These types of interfaces are also similar to the one employed by Clearpath Robotics as part of their OTTO warehouse-transport system. The OTTO interface enables operators to control robots by creating various constraints on their behaviours (e.g. directional constraints, speed constraints, stop signs, etc.) covering specific areas of the map. In current industrial practice such rules for robot behaviour are designed by trained personnel [73].

4.3 Interface Implementation

Considering the properties of effective interfaces identified by [66, 67, 68], and existing interfaces for commercial systems [69, 70, 71, 72], we now describe the implementation of our robotics interface. We chose to develop our interface using the ROS ecosystem, with the interface consisting of a plugin for the Robot Visualisation (RVIZ) package, which provided two major advantages: single interface to be used on physical and simulated robots, and ability to use various packages in the ecosystem to easily generalise to different robot platforms and environments. Implementing the base Floormap Visualization that is underlying the rest of our interface was made simple due to the existing ecosystem, which already contained occupancy grid viewing capabilities. Existing functionality also allows for the drawing of "marker" elements on the map (e.g. lines, arrows, points).

Moving on to the Specification Phase interface, overall its design is very similar to the OTTO interface. Our interface (seen in Figure 4.1) is split into two sections: tag creation/overview panel, and the map-view panel. The map-view panel presents an operator

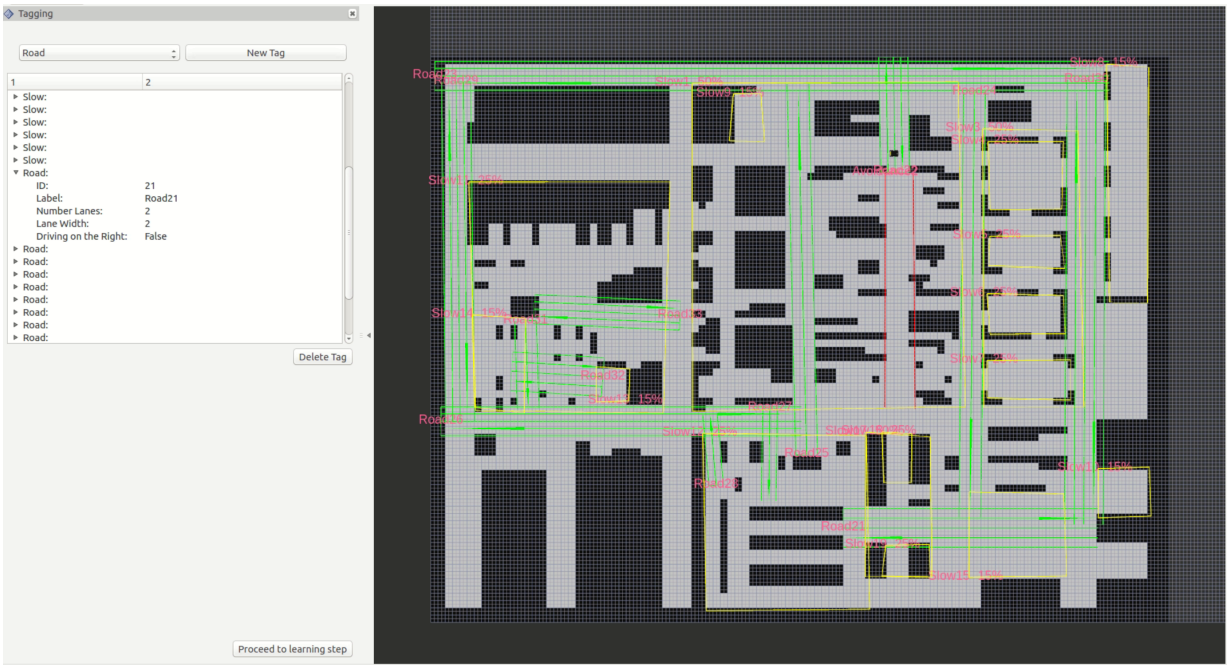


Figure 4.1: GUI of Specification Phase

with an occupancy grid floor map of the environment, that encodes free and occupied environment space. The tag creation/overview panel allows the operator to create behaviour constraints by selecting the constraint type, filling in any required metadata, and then designating its location and area on the map-view. Additionally, this panel also provides an overview of all existing constraints. Examples of constraints include directional constraints (e.g. roads, where movement is allowed in one direction, but not the opposite), temporal constraints (e.g. stop signs), speed constraints (e.g reduced speed zones), or positional constraints (e.g. restricted zones). This UI design results in the full interface only taking up a single window, with no portions of the UI occluded by other parts of it, decreasing the workload involved in keeping track of various aspects of the system.

Once the user is satisfied with their initial specification, they can proceed to the Learning Phase. The Learning Phase interface (Figure 4.2) maintains the two panel arrangement, however the tag creation/overview panel is replaced by the learning panel. The learning panel presents the operator with two pieces of information about the paths to be compared: time duration, and list of violated constraints. The paths are also added to the map-view panel, and the operator can highlight each path (and their violated constraints) from the learning panel. Based on the information provided about each path, the operator makes

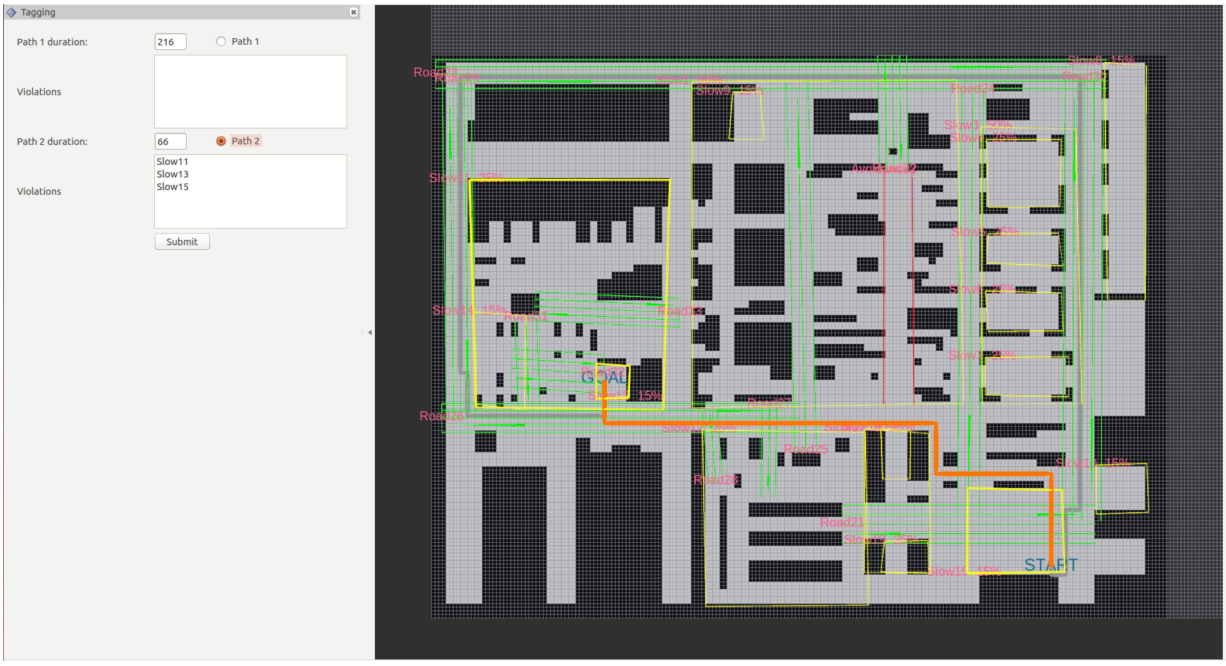


Figure 4.2: GUI of Learning Phase

their preference known to the system. Maintaining the same map-view panel from the Specification Phase, ensures that the user is already accustomed with the interface and will not need time to get reacquainted.

The third component of the system is the Questionnaire Phase, which presents the user with a pop-up window directing them to answer several sets of questions. For example, Figure 4.3 depicts one of the post-study set of questions presented to the participants in the study conducted in Chapter 6. The Questionnaire Phase interface can also be invoked throughout the Specification and Learning phase to ask the user any specific Specification/Learning questions. For example, in our study implementations, we have several questions that are asked once the user finishes each of the Specification and Learning phases.

Having described the interface employed in obtaining user specifications (and revisions if applicable), we now turn our attention towards developing a set of metrics to assess the quality of a specification, which is the focus of the next chapter. These metrics are used on the initial specification, as well as on revised specifications obtained through the active learning process.

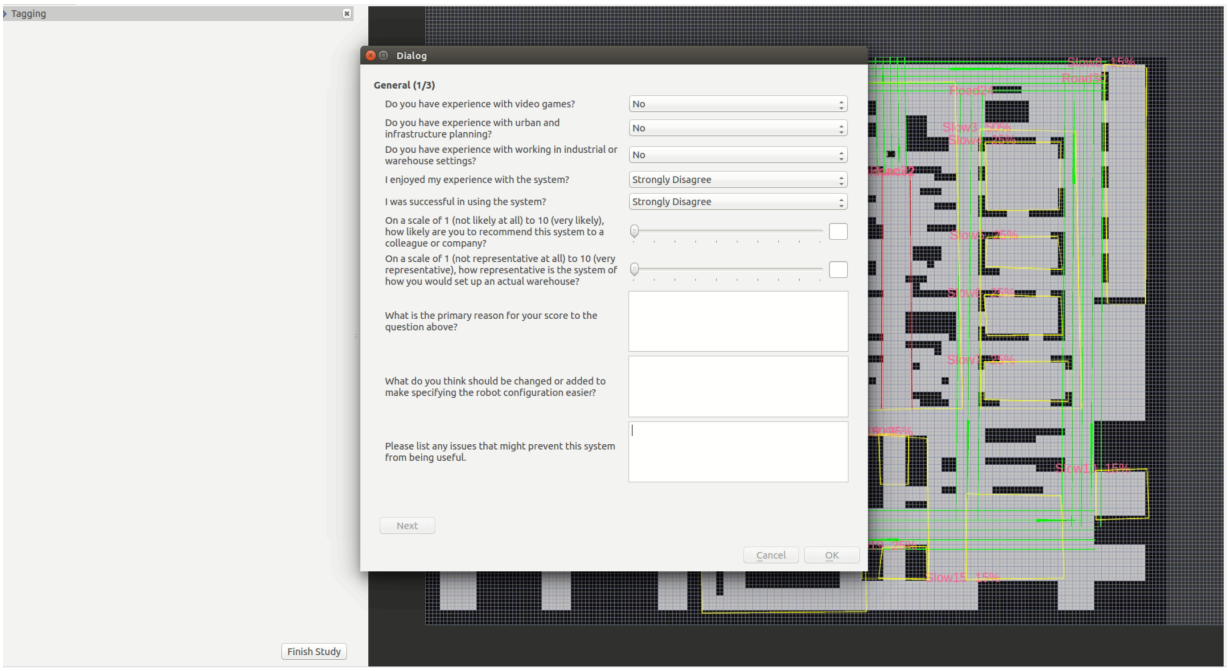


Figure 4.3: GUI of Questionnaire Phase

Chapter 5

Metrics Development

This chapter[1] describes the initial development of metrics to be used in quantifying user specifications. Towards that goal, we first introduce and discuss the metrics capturing the positive and negative effects of specifications. This is followed by a description of a preliminary user study focused on obtaining a set of user-created specification that would be used to test the proposed metrics. The final sections analyze the results of the preliminary study and summarize the findings used to guide the subsequent user study described in Chapter [ref].

5.1 Metrics Description

We now describe the metrics used to evaluate the quality of a given specification, which is the set of user-created behavioural constraints. We propose three metrics: Entropy, Shortest Paths Coverage Area (`SPCA`), and Loss of Efficiency (`LoE`). The first two metrics capture the positive effects of a specification, related to its ability to constrain robot behaviour to be more predictable and better conform to user expectations. The last metric, `LoE`, captures the negative effects related to the performance losses encountered by the robot (in particular, the increase in distance traveled by the robot), due to the limits placed on its behaviour. These metrics can be computed on any motion graphs that represent a robot's potential movement in the environment.

[1] A version of this chapter has been submitted in [8]

5.1.1 Preliminaries

Consider motion graph $G = (V, E, \Psi)$, where the function $\Psi : E \to \{(v, w) \in V \times V : v \neq w\}$ associates each edge with an ordered pair of vertices. Given a vertex v we call a vertex w a neighbour of v if v is the start and w the endpoint of an edge in G. We denote the set of all neighbours of v as $\mathcal{N}(v)$.

When a constraint is created, the costs of the appropriate edges covered by the constraint are affected. For example, consider the motion graph in Figure 5.1, where a Road zone is placed from x_3 to x_1, and we set the Speedup factor to be equal to 2. This results in decreasing the cost of the edges directed along the Road direction by the Speedup factor, while edges directed in the opposite direction get deleted by having their cost set to ∞. Edges whose vertices are not both within the Road zone get their costs increased by the Speedup factor, discouraging the robot from leaving the Road too early.

5.1.2 Positive Effects

The primary intent of a user specification is to constrain the robot behaviour to make the robot's actions more predictable, and better align them with a user's expectations.

In the context of warehouse robotics, increasing robot predictability primarily refers to making it easier for workers to determine the paths that a robot will take during operation. This applies to both operator (supervisor) interactions during tag creation, and to interactions the robot will have with warehouse staff (bystanders) while it navigates the warehouse to accomplish its goals.

A second intent of a specification is to align the behaviour of the robot with user expectations, which has a positive effect on the usage and effectiveness rate of the robots. This is an important consideration, and in many cases, the existing alternatives to autonomous mobile robots (i.e. AGVs and forklifts) are also typically constrained to operate on fixed roads or to obey traffic rules, such that their behaviour generally matches user expectation.

Entropy

Yang and Anderson [74] proposed to use entropy to measure the complexity of an environment, by quantifying the number of decisions that a robot has to make to traverse the environment. We propose to use the environmental entropy to quantify how a specification affects the complexity of the robot's decision making. Following Yang and Anderson [74], the entropy of a given node in the motion graph can be calculated using

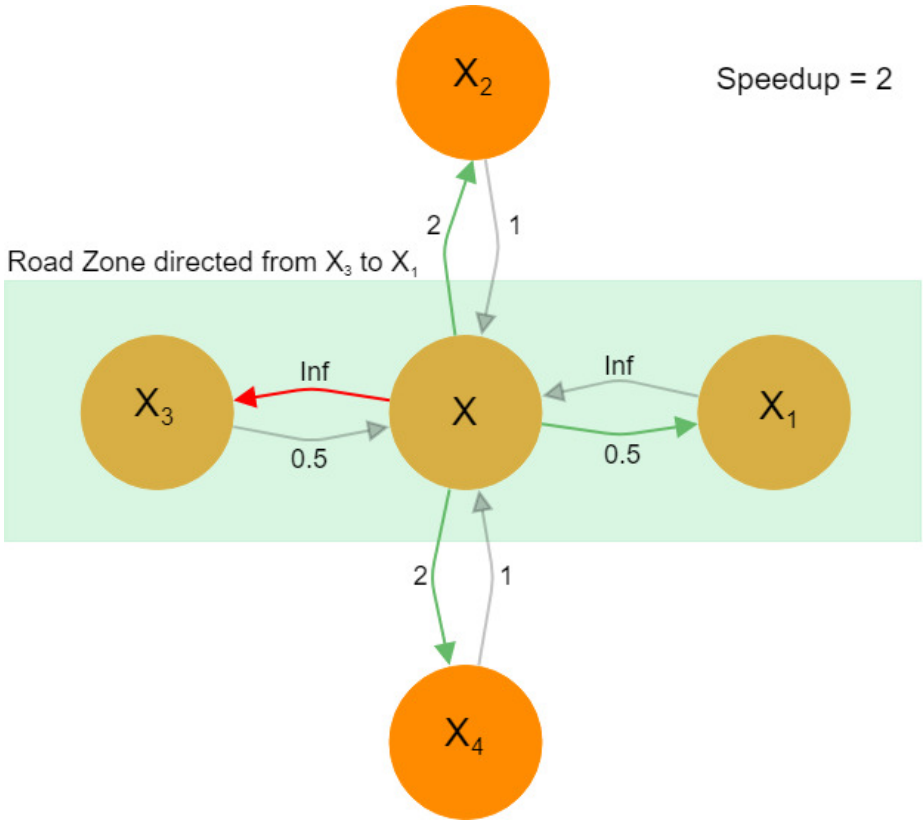


Figure 5.1: Examples motion graph generation for a vertex in a *Road* zone from left to right. The edges modified at vertex X are in color. The edge between X and X_3 is going against the *Road* direction and is deleted (cost becomes infinite, highlighted in red). Edges between X and X_2, and X and X_4 are edges leaving the *Road*; their cost is increased (highlighted in green). The edge between X and X_1 is fully contained inside the *Road*, and it is in the forward direction so its cost is decreased (highlighted in green).

$$H(x_i) = -\sum_{j \in \mathcal{N}(x_i)} p(x_i, x_j) \log_2 p(x_i, x_j),$$

where given a node x_i, x_j is a neighbor of x_i, and $p(x_i, x_j)$ is the "probability" of the robot transitioning from node x_i to node x_j, i.e., traversing the edge (x_i, x_j). The transition probability captures a localized view of the graph, as it assumes that at a first approximation, transition decisions are based on the edge weights of each outgoing edge from a given vertex. The following formula is used to calculate the transition probability corresponding to each edge

$$p(x_i, x_j) = \frac{\frac{1}{W(x_i, x_j)}}{\sum_{k \in \mathcal{N}(i)} \frac{1}{W(x_i, x_k)}},$$

where $p(x_i, x_j)$ is the probability associated with transitioning along the edge from x_i to x_j, $W(x_i, x_j)$ is the weight of the edge (x_i, x_j), and $W(x_i, x_k)$ is the weight of the edge (x_i, x_k), where x_k is a neighbour of x_i.

For nodes that have no neighbors (outdegree of 0), the entropy is zero, while for a node where all outedges have an equal weight, the probability to move to any of them will be equal, and the entropy of that node will be equal to 2. When the outedges do not all have the same weights, entropy is reduced as the robot is more likely to traverse the higher probability edge. As an example, consider the set of nodes presented in Fig. 5.1, where we are interested in calculating the entropy of node x. In Fig. 5.1, nodes x_3, x, and x_1 are part of a *Road* zone directed from left to right. Applying this formula results in $p(x, x_1)$ being equal to $0.\overline{6}$, while $p(x, x_2)$ and $p(x, x_4)$ are equal to $0.1\overline{6}$ and $p(x, x_3)$ is equal to 0.

To compute the overall entropy of the environment, the entropies of all nodes are summed:

$$H = \sum_{i=0}^{n} H(x_i),$$

where n is the number of nodes in the graph, $H(x_i)$ is the entropy of each node in the graph, and H is the total entropy of the graph.

Shortest Paths Coverage Area

The second metric that captures the positive impact of constraints in a specification is related to the area covered by the shortest paths with the minimum number of turns between start and goal points.

In an environment with no restrictions, there will generally be a large number of candidate shortest paths between two points. Adding restrictions on robot behaviour, however, limits the number of shortest paths in the environment. By reducing the number of viable paths between the start and end points, the area covered by those paths is then also potentially reduced. A simple example are two rooms in an environment connected by two separate hallways, resulting in two different paths between the rooms. Placing a *No-go* zone on one of the hallways will result in only one path remaining between the two rooms.

There are two ways that the coverage area can be calculated and interpreted. The first is to count the number of nodes that are traversed by any of the shortest paths. For this, we define S and G as the set of start and goal nodes, respectively. We then define the following indicator function:

$$I(x_i) = \begin{cases} 1, & \text{if } x_i \in P(s,g) \text{ for some } s \in S, \text{ and } g \in G, \\ 0, & \text{otherwise,} \end{cases}$$

which outputs 1 if the node x_i is part of any of the shortest paths $P(s,g)$. To obtain the shortest paths coverage area, `SPCA`, of a specification, we sum up the result of the indicator function for each of the n nodes of the environment:

$$\texttt{SPCA} = \sum_{i=1}^{n} I(x_i).$$

The shortest paths in the graph are found using Dijkstra's algorithm.

In the second approach, we first compute all shortest paths for each $s \in S$ and $g \in G$. Let the number of such paths be $m(s,g)$. Then the total number of shortest paths over all tasks is $M = \sum_{s \in S, g \in G} m(s,g)$. Then, for each node x_i, we let $C(x_i)$ be the number of those M shortest paths that pass through node x_i. Finally, we define

$$A(x_i) = \frac{C(x_i)}{M}.$$

This quantity gives the fraction of shortest paths from S to G that pass through x_i.

It is important to highlight that a robot will not necessarily pass through all of the areas indicated by the shortest paths coverage, as the shortest paths coverage area is the result of the union of individual paths, and the robot will only navigate on one of those paths. We consider all possible shortest paths in the `SPCA` metric, since selecting among the shortest paths is an additional optimization problem outside the scope of this work.

5.1.3 Negative Effects

The negative effects of a specification aim to capture the drawbacks of constraining robot behaviour. The major drawback of the user constraints comes from increasing the length of shortest paths between start and end points compared to the baseline motion graph.

Loss of Efficiency

The `LoE` metric seeks to determine the relative increase in length of the shortest path between a pair of points on the map between a motion graph obtained from a specification and the baseline motion graph, using the following formula:

$$\texttt{LoE}(x_i, x_j) = \frac{d_s(x_i, x_j) - d_b(x_i, x_j)}{d_b(x_i, x_j)},$$

where x_i and x_j are two nodes of the graph, $d_s(x_i, x_j)$ is the distance between x_i and x_j on the specification motion graph, and $d_b(x_i, x_j)$ is the distance between the two nodes on the baseline motion graph with no specifications applied. Where there is no path between two nodes, the distance between them is set to be equal to n, the total number of nodes in the graph, when calculating the Global `LoE`, and `Inf` when calculating the task `LoE`. This calculation can be applied to a single task (a single pair of start and end positions), a set of tasks, or to all pairs of nodes in the graph. Based on this metric, we can then calculate the mean `LoE` of the entire specification.

5.2 Study Design

Following the development of the metrics, we now describe the design of a preliminary study that would allow us to obtain user behaviour specifications, which would then be used to test the previously introduced metrics. In this section we will first describe the study scenario presented to participants, and then the study procedure.

5.2.1 Scenario Description

First, we describe the scenario employed in the study used to motivate the user in creating a user specification. The scenario consists of a warehouse setting, where a robot is tasked

with completing a simulated material transport tasks, which is to say that the robot will move from a starting location to a goal location in the environment.

To begin the specification process, the user is provided with a floor plan of the scenario environment, reproduced in Fig. 5.2, and a written description of the various elements of the environment. As part of this description, it is indicated that there are three main sections to the floor plan: the shelves area at the top, the central area, and the shelves area at the bottom. The description continues to say that the two shelves areas contain a grid of roads with wide vertical corridors that permit traffic in both directions, and a narrow horizontal corridors where only a single direction of traffic can exist. Both shelves areas are already used by forklifts, and as such pedestrian workers are used to working alongside heavy machinery. Additionally, the bottom shelving area contains a Tech Lab, which doesn't allow for any heave machinery traffic due tot he nature of the items inside.

The central area is is described as featuring three sub-sections. The first two are the side corridors, which link the two shelving area. These hallways are used primarily by forklifts and other heavy machinery to move items between the two sides of the warehouse. The third sub-section is the large opening in the middle. This area acts as a lobby and rest area, and as a result it is much more populated by warehouse employees and other visitors. This area also includes stationary elements including desks, chairs, vending machine, and cupboards. Workers and visitors will only be present in this area during the day, and evening shifts.

Additionally the three potential starting points, and three potential end points of the material transport task are shown with green, and red circles respectively. Once introduced to the environment, the robot will be required to move from any one starting point, to any of the three end points.

5.2.2 Study Procedure

The study as described below was approved by the Office of Research Ethics at the University of Waterloo (ORE #22167).

Each study session took approximately 1 hour. During this time, participants were asked to sign a consent form, after which they were briefed with an introduction and overview of the study, and their tasks as participants. The participants then proceeded consecutively through the following three stages: training, experiment, and post-study questionnaire.

During the training stage, participants were given the chance to familiarize themselves

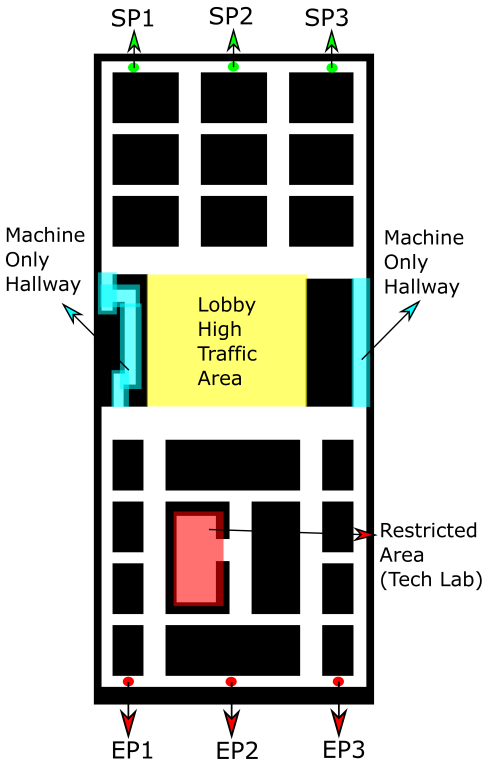


Figure 5.2: Map of the target environment. Empty space is in white and occupied space is in black. The green dots represent the 3 possible starting locations while the red dots represent the 3 possible end locations. The central empty area is the "Lobby" of the warehouse, while the two hallways to the left and right of the "Lobby" are machine only areas.

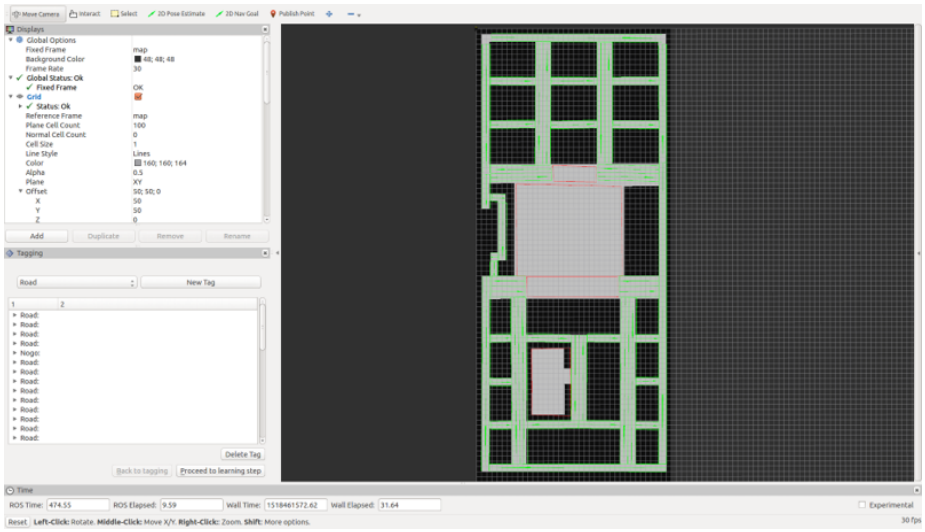


Figure 5.3: RVIZ-based interface used in the study, showing an environment bearing a large amount of constraints. Road constraints are shown in green, No-go in red, Preference in yellow, and Targets in blue.

with the interface, and were introduced to RVIZ, Gazebo, and the constraints at their disposal. They were then provided with hands-on time with the interface, being instructed to create at at least one of each of the available constraints. Following that, the participants were then asked to familiarize themselves with tele-operating the robot through the simplified training environment. Participants were not given any time limits for their training, and so it was up to them to inform us when they were ready to conclude their training.

During the experiment stage, participants were given a floor plan of the environment (reproduced in Fig. 5.2), as well as a written scenario description intended to provide context on the various areas of the map. The participants were then instructed to use this information to create constraints that would ensure the robot could safely reach its intended goals, from any of the three starting locations. Furthermore, it was indicated that they could use as many constraints as they wanted, and that they should continue the specification process until they were satisfied with their efforts. Once participants indicated their satisfaction, they were then asked to tele-operate the robot from $SP1$ to $EP2$ in Fig. 5.2. The experiment stage of the study was terminated when the robot successfully arrived at the $EP2$ location.

Finally, following the conclusion of the experiment stage, participants were asked to fill out a questionnaire, and optionally expand on their answers verbally. The questionnaire

Table 5.1: Questions contained in the questionnaire. To not overwhelm participants, only one group of questions was shown at a given time.

Group	Question	Rating Scale	Qualitative Description
A	How straightforward did you find the system to be?	1 → 10	Not straightforward at all → Very straightforward
	How CONFIDENT were you in using the zone tagging environment?	1 → 10	Not confident at all → Very confident
	How CONFIDENT were you in tagging the environment correctly?	1 → 10	Not confident at all → Very confident
	How CONFIDENT were you in tele-operating the robot?	1 → 10	Not confident at all → Very confident
B	How WELL do you think you performed in using the zone tagging environment?	1 → 10	Performed very poorly → Performed very well
	How WELL do you think you performed in tagging the environment correctly?	1 → 10	Performed very poorly → Performed very well
	How WELL do you think you performed in tele-operating the robot?	1 → 10	Performed very poorly → Performed very well
	How WELL do you think you specified the robot task?	1 → 10	Poor specification → Excellent specification
C	Prior to the study, did you have any relevant video game experience?	Yes/No	
	When you tele-operated the robot, did you follow the specification you created?	Yes/No	
	Why or why not?	Text	
D	I think that I would like to use this system frequently.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I found the system unnecessarily complex.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I thought the system was easy to use.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I think that I would need the support of a technical person to be able to use this system.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I found the various functions in this system were well integrated.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I thought there was too much inconsistency in this system.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I would imagine that most people would learn to use this system very quickly.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I found the system very cumbersome to use.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I felt very confident using the system.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree
	I needed to learn a lot of things before I could get going with this system.	Likert Scale	Strongly disagree / Disagree / Neither agree nor disagree / Agree / Strongly agree

was split into two sets of questions. The first set asked the participants to assess their own performance during the study, while the second set contained the standardized System Usability Scale (SUS), which are questions regarding the usability of the system and the ease/difficulty of using the developed interface. Both set of questions are shown in Table 5.1.

5.2.3 Hypothesis

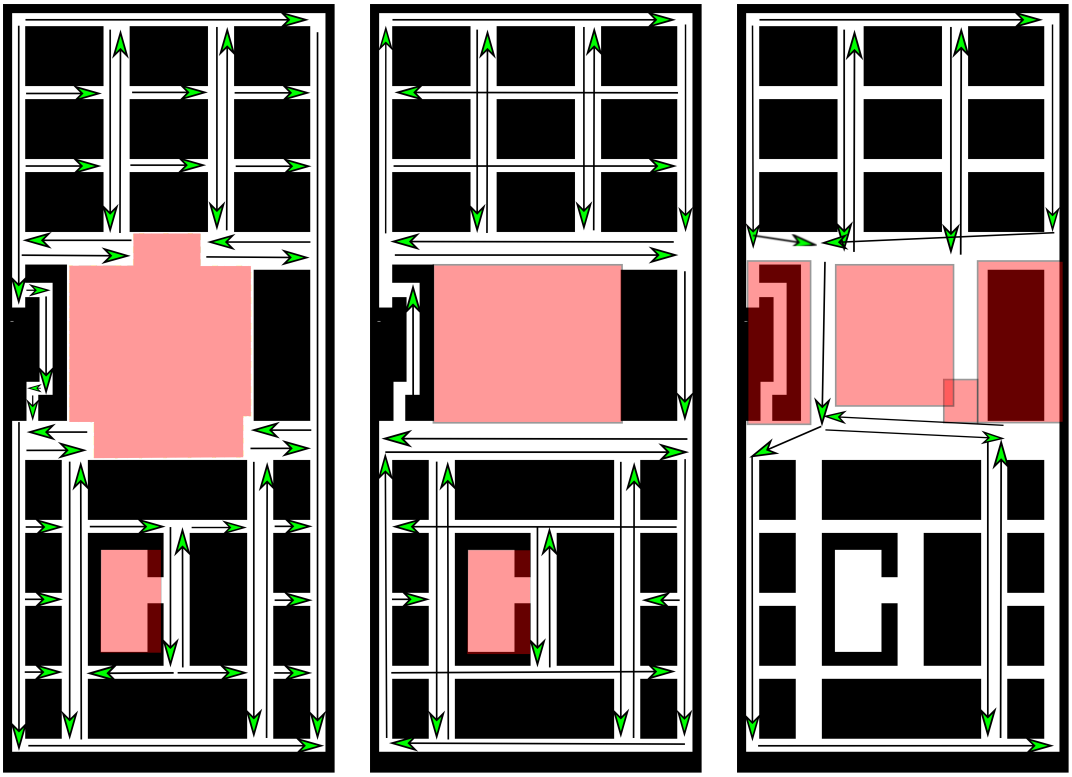
We proposed the following hypotheses for the preliminary user study:

Hypothesis 1 (H1) *Users create specifications achieving varying performance scores.*

To validate Hypothesis 1, we apply the proposed metrics to the user-created specifications, identifying which users over constrain robot behaviour and fall short of achieving performant specifications.

5.3 Results

As part of this study, we recruited 8 University of Waterloo students by email. Out of these participants, 6 were undergraduate students, while 2 were graduate students. No other user demographics data was collected.



(a) Specification 2 (b) Specification 4 (c) Specification 5

Figure 5.4: Sample user specifications

5.3.1 Specification Variety

Even though the participants were provided with identical task instructions, a large variety of specifications were observed, both in number of tags and in the area that they occupied. The average number of tags contained in a specification was 23.12, with a standard deviation $\sigma = 11.31$, and on average, tags covered 53.48% of the map ($\sigma = 16.38\%$).

In addition to variations in the number and area coverage of tags, there was also a large variety in terms of how the tags are configured and specified. In the example specifications in Fig. 5.4, we can see that Roads differ in direction, coverage and the amount of overlap between different Road zones. Similarly, the extents and locations for No-Go zones differ between specifications.

The specifications indicated that the extent to which participants considered *global*

Table 5.2: Mean and standard deviation values for each of the performance metrics

SPCA	Entropy	Task LoE	Global LoE
190 ($\sigma = 100$)	551 ($\sigma = 169$)	16 ($\sigma = 9$)	4172 ($\sigma = 1795$)

performance (i.e., performance outside of the specified start and end positions) also differed. The global performance of a specification would be useful if the robot were to perform tasks other than the ones described in the study, or if the robot was disturbed and then had to re-plan from an unexpected area of the environment. One example of differences between the participants in terms of global considerations comes in the form of the *No-Go* tags that some specifications have on the Lab Tech room, despite the fact that based on the robot's given task set, it should have no reason to navigate through that room.

The different specifications resulted in considerable differences in the performance levels, as seen in Table 5.2, supporting Hypothesis 1. Note the mean and standard deviation calculation for the Task Average LoE does not include the outlier specification observed in Fig. 5.5 for reasons detailed in Section 5.3.2.

5.3.2 Task-specific Performance

We first analyze how participants traded-off between positive and negative effects for task-specific performance, i.e., when only the performance on the specified task is considered. The relevant metrics are the Task SPCA, illustrated in Fig. 5.7, and Task Average LoE (increase in distance travelled for task completion).

Fig. 5.5 immediately indicates one large outlier, Specification 2. In this specification, the robot cannot reach all of the end points from all of the starting points (e.g. EP3 from SP3) which results in a score of infinity. Our LoE metric was able to identify the failure of Specification 2, which was not immediately obvious by looking at Fig. 5.4a. The rest of the specifications averaged a task LoE score of 17%, with three of them attaining a task LoE score of under 10%. It should be noted that the three specifications with a low LoE score have higher SPCA scores than the rest. In general, specifications that have a larger SPCA have smaller efficiency losses.

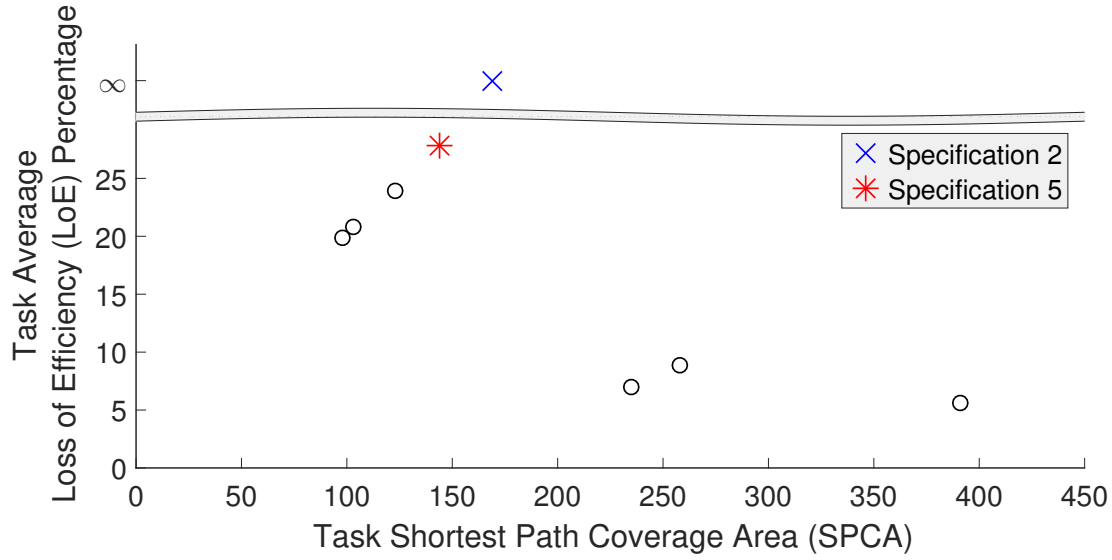


Figure 5.5: Average `LoE` versus `SPCA`. In this figure, the average `LoE` was calculated over the 9 pair of start and end points that specified the robot task.

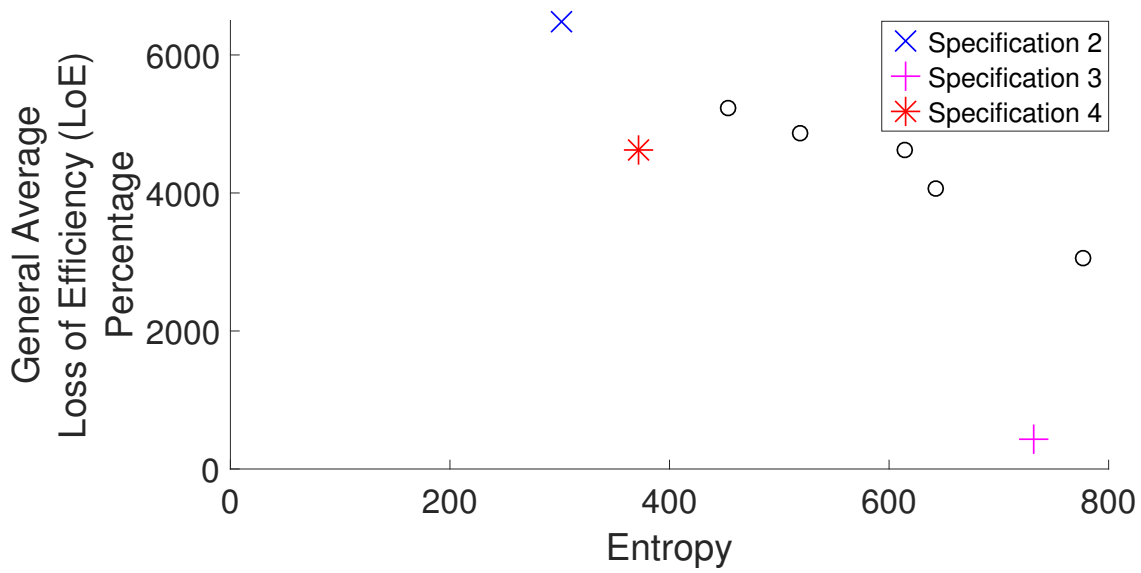


Figure 5.6: Average `LoE` versus Entropy. In this figure, Average `LoE` was calculated over all points in the map.

5.3.3 Global Performance

Although participants were told of only one set of tasks that the robot needs to accomplish, it is useful to examine how performance is impacted if a robot was required to solve additional tasks using the same specification, or if disturbances forced the robot away from paths connecting the task start and end points. Since we are interested in task agnostic performance, Entropy and Global Average `LoE` will be used as the metrics representing the positive and negative effects of a specification globally.

While a few participants created specifications with low levels of Task Average `LoE`, that was not the case for General Average `LoE`. As Fig. 5.6 indicates, only one specification obtained a relatively low Global `LoE` score(430%), while the rest obtained much higher scores (average 4700%). This leads us to believe that participants did not generally consider the global implications of their specifications, and that they primarily constructed them to serve the sample task provided in the study. This is further evidenced by observing that some specifications (e.g Specification 5) completely block the flow of traffic from the lower side of the map to the top side in Fig. 5.2, which results in high efficiency losses.

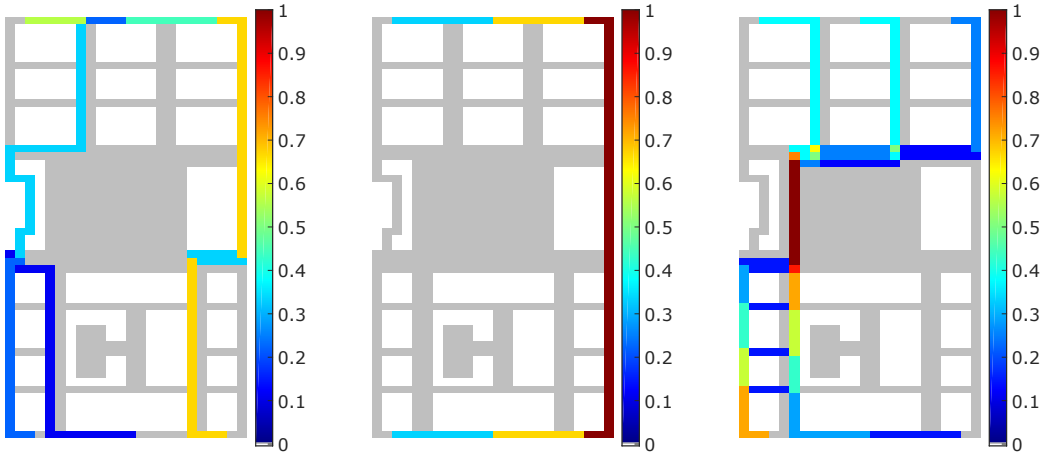
At the global level, it can be difficult for a human operator to predict how even small changes in the specification can affect the performance of the robot. This is seen in Fig. 5.6 with Specifications 2 and 4, where despite having similar entropy levels (predictability of behaviour), there is a very large difference in their Global `LoE` values, which may not be apparent by simply looking at the specifications (Fig. 5.4).

5.3.4 User Questionnaire

After completing the specification, the participants were asked to assess their task performance.

The responses, comparing user assessment of their performance at specifying the robot task against Task `LoE`, is shown in Fig. 5.8. Surprisingly, the three participants who created the specifications with the smallest Task `LoE` scores, and thus the fewest negative effects on performance, rated their performance below the average of the participants who authored the high Task `LoE` specifications (7.3 versus 8.8 rating). While the participant who created a specification that failed to meet the task objectives (Specification 2), rated their performance slightly under the average rating of the top performers, their rating was still high, despite failing to meet the robot task.

Since the users' self-assessment ratings do not match their specification's Task `LoE` performance, it was hypothesized that the users might have instead rated the global per-



(a) Specification 2 (b) Specification 4 (c) Specification 5

Figure 5.7: Shortest Paths Coverage Area of 3 specs, illustrating the differences between specs. The colors encode the ratio of shortest paths that pass through each point on the map

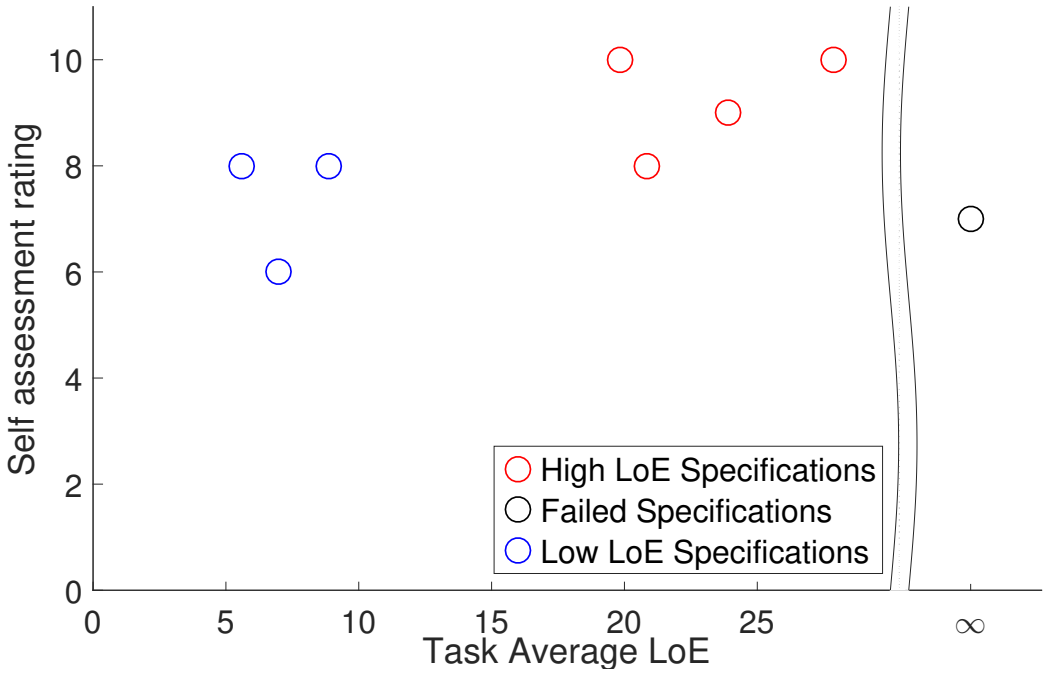


Figure 5.8: The participants' self-assessed rating of how well they specified the robot task, shown against the Task Average `LoE` scores of their specifications

formance of their specifications. However, upon investigating how the users' self-assessment rating matches the Global `LoE`, we identified a similar relationship to that in Fig. 5.8. That is, participants with a lower Global `LoE` rated themselves lower than participants with a higher Global `LoE`. This indicates that it was difficult for participants to accurately rate the performance of their specifications.

5.4 Discussion

Our results found that the choice of tags can have a large impact on the performance and usability of the robot system, and that novice users who are unfamiliar with the specification system may not be able to fully appreciate the impact of specification choices on subsequent robot performance, as even small changes in the specification can lead to different robot behaviour. An extreme example of this was seen with the one participant who was unaware that they had created a specification that prevented the robot from accomplishing its tasks. Our metrics were able to identify this failure. This indicates that they could be used as a verification tool during the specification design process. However, these results needed to be confirmed through studies with a larger number of participants.

These findings motivated the need for an interface that can guide users towards obtaining better specifications. Therefore, in the next stage of the research, described in Chapter 6 we integrated the performance metrics with a preference learning system similar to [53, 75]. Such a system would allow us to better align the behaviour of the robot with user expectations while minimizing performance loss, based on the feedback obtained from the user. In addition, it would also allow the user to better understand the effects that their configuration has on robot performance, and improve their ability to accurately rate their performance.

Chapter 6

Interactive Preference Learning User Study

The work presented in this chapter[1] builds upon the work presented in Chapter 5, and in [53] (summarised in Chapter 3). In this chapter we present a system that integrates a specification-creation interface with a learning approach that iteratively improves the robot performance by modifying the specification based on user interaction. We then describe the design of a user study that tests the proposed approach. The final sections analyze the results of the user study, and summarize its findings.

6.1 Motivation

Besides showing the capabilities of the chosen metrics in describing the quality of a specification, the preliminary study also showed the difficulties participants faced in creating efficient specifications, as well as the difficulty in accurately rating their own performance. We also observed these difficulties at one of Clearpath Robotics' OTTO training sessions for robotics integrator personnel that I attended in the summer of 2017. The training session is intended to teach industrial domain experts how to properly and efficiently set up and deploy the OTTO robots in industrial facilities. The interface that Clearpath uses for the OTTO robots inspired our interface: it allows an operator to create area-based behaviour constraints on robot behaviour. Beyond constraint-specific metadata, each constraint also has one or more Weight data fields, which operators manually assign values

[1] A version of this chapter has been submitted in [9]

to. By trial-and-error, the operator can then specify the behaviour that they desire from their robots by modulating the weights of the constraints. Through our informal observation and discussion with participants and instructors, accurately setting weights to obtain the desired behaviour was observed to be a time consuming process requiring significant training.

To reduce the challenge for the operator and improve robot performance, in this chapter we combine our preliminary interaction system with the preference learning system in [53], incorporating the interaction/interface guidelines presented in Section 4. The preference learning process is hypothesized to help in several ways: First, it reduces the disparity between the robot's actual behaviour, and the operator's expectation of how the robot should behave, as the combined system would ask the user which behaviour they preferred, and would automatically modify the underlying weights accordingly. Second, the combined system would appropriately take advantage of the operator's and robot's skills and capabilities, as the human operator is better able to understand the requirements of the human world, while the robot is much better at determining where its operation is inefficient and could be improved. Finally, the combined system would avoid the training time spent in learning the weight-to-behaviour connection, as well as the time spent performing trial-and-error modifications to the weights with each new deployment.

6.2 Proposed Approach

6.2.1 Preliminaries

Using definitions from [76], a multi-graph is a triple $G = (V, E, \Psi)$, where the function $\Psi : E \to \{(v, w) \in V \times V : v \neq w\}$ associates each edge with an ordered pair of vertices. Given a vertex v we call a vertex w a neighbour of v if v is the start and w the endpoint of an edge in G. We denote the set of all neighbours of v as $\mathcal{N}(v)$. Multiple edges are allowed to connect the same ordered pair of vertices and are then called parallel. In our problem we consider doubly weighed multi-graphs of the form $G = (V, E, \Psi, c_1, c_2)$, where c_1 and c_2 are independent weight functions, each associating a real number to each edge of the graph: $c_i : E(G) \to \mathbb{R}$ for $i \in \{1, 2\}$.

A walk between two vertices v_1 and v_{k+1} on a graph G is a finite sequence of vertices and edges $v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1}$ where $e_1, e_2, \ldots e_k$ are distinct. A path $P_{v_1, v_{k+1}}$ between two vertices v_1 and v_{k+1} is defined as a graph $(\{v_1, v_2, \ldots, v_{k+1}\}, \{e_1, e_2, \ldots, e_k\})$ where $v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1}$ is a walk. On a weighted graph, the cost of a path is defined

as $c(P) = \sum_{e \in P} c(e)$. In doubly weighted graphs we define two costs c_1 and c_2 where $c_1(P) = \sum_{e \in P} c_1(e),\ c_2(P) = \sum_{e \in P} c_2(e)$.

Notation Vectors are written with bold, lower case letters, e.g., $\boldsymbol{v}$, we address elements of the vector with a subscript index v_i. A superscript index $\boldsymbol{v}^i$ identifies a specific vector. Sets are denoted by upper case letters (G), matrices as bold upper case letters ($\boldsymbol{A}$).

6.2.2 Problem description

The proposed approach contains the following two components: First, having obtained a user specification, we use an extension of the active learning technique introduced in [1], to gain information about the importance of the user constraints, i.e., the user's preference between alternative paths. The technique is extended to allow its use with a multi-task scenario. Following this, we apply the metrics proposed in [8] to evaluate the impact of the specification, and show how the learning system improves the quality of the robot's task performance.

6.2.3 Learning user preferences

Problem setup

The learning system receives a description of the environment, the user specification and a set of tasks. The environment is considered to be static and is represented as a weighted strongly connected multigraph $G = (V, E, \Psi, t)$. The weight t on the graph encodes the time a robot requires to traverse an edge. We use parallel edges with different times to model speed. We extend our previous work from [1] and consider a set of ordered pairs $\{(s_1, g_1), (s_2, g_2), \dots\}$ where s_i and g_i are vertices on G. A single task consists of navigating from a start s_i to a goal g_i. On the environment map, the user specifies a set of constraints $\Gamma = \{\gamma_1, \gamma_2 \dots, \gamma_d\}$. Each constraint γ_k is a pair (E_k, w_k^*), where E_k is a subset of the edges of G and w_k^* is a hidden user cost for the constraint. Notice that a road on the interface entails two constraints: A reward for using the road in the direction of travel and a penalty for moving the wrong way. Consequently, a two way road maps to four constraints. To incorporate the user specification, we create a doubly weighted graph $G^\Gamma = (V, E, \Psi, t, w^*)$. For each edge e in G^Γ the second weight $w^*(e)$ is defined as the sum of all w_k^* that belong

to a constraint containing e. Our objective is to find paths P_i^* between all s_i and g_i that are optimal with respect to

$$\min_{P_i} \sum_{e \in P_i} w^*(e) + t(e). \tag{6.1}$$

The true user weights w_k^* are latent, i.e., we do not ask the user to define w_k^* during the specification. Nonetheless, given estimates $\hat{w}_k$ of the true user weights, we can also construct a doubly weighted multigraph $\hat{G}^\Gamma$. Moreover, the weights are defined in units of time, allowing us to pose the multi-objective optimization as an unweighted sum. To learn about the weights, we can query the user. In a query we present them with a pair of paths (P_i^1, P_i^2) for a selected start-goal pair (s_i, g_i). Considering only pairs instead of more than two paths at a time is motivated by reducing the burden on the user, as choosing between numerous alternatives is more demanding [77].

Linear learning model

Given a specification of d constraints, the latent user weights can be be summarized as a column vector $\boldsymbol{w}^* \in \mathbb{R}^d_{\geq 0}$. Furthermore, a path P is described by the time it takes to traverse $t(P)$ and a vector $\boldsymbol{\phi} \in \mathbb{Z}^d$ that indicates for each constraint $\gamma_k \in \Gamma$ how many edges in E_k are traversed by a path, i.e., $\phi_k(P) = |E(P) \cap E_k|$. The cost of a path is then written as $C(P) = \boldsymbol{\phi}(P) \cdot \boldsymbol{w}^* + t(P)$.
In our terminology we distinguish *penalty* and *reward* constraints. Penalty constraints include the edges within an avoid zone, edges within a speed-limit zone where the traversal time does not correspond to obeying the speed limit and the edges going against the defined direction of travel in a one-way road. Reward constraints describe the edges that follow the direction of traffic on a road. In our convention, ϕ_i takes positive values for penalty constraints and negative values for reward constraints.

As $\boldsymbol{w}^*$ is hidden, we initially only know that it is positive and finite; hence $0 \leq w_i^* \leq w_i^{\max}$ for sufficiently large values $w_i^{\max}$. For the penalty constraints, $w_i^{\max}$ is set to the sum of all t_i for all edges e_i on the graph G. On the other hand, the rewards for following roads require a tight upper bound to avoid negative cycles. Let γ_i be a constraint containing edges that follow a road. To obtain the upper bound we choose the length of the shortest edge in the constraint, denoted by $t_i^{\min}$. Further, we subtract a small discount such that a path planner breaks ties in favor of paths using fewer edges: $w_i^{\max} = (1 - \epsilon) t_i^{\min}$ where $0 < \epsilon \ll 1$.

Let P^i and P^j be two paths. If the user prefers path P^i it implies that $C(P^i) \leq C(P^j)$.

We can write this as a half-space in $\mathbb{R}^d$ containing $\boldsymbol{w}^*$

$$\{\boldsymbol{w} \in \mathbb{R}^d_{\geq 0} | (\boldsymbol{\phi}^i - \boldsymbol{\phi}^j)\boldsymbol{w}^* \leq t^j - t^i\}. \tag{6.2}$$

Thus, obtaining user feedback allows us to iteratively learn inequality constraints on the user weights. We write the intersection of the learned half-spaces as a polyhedron

$$\mathcal{F} = \{\boldsymbol{w} \in \mathbb{R}^d_{\geq 0} | 0 \leq w_i \leq w_i^{\max},\ \boldsymbol{A}\boldsymbol{w}^* \leq \boldsymbol{b}\}, \tag{6.3}$$

which we refer to as the *feasible space*. In [1] we have shown that the feasible space shrinks whenever new user feedback is obtained.

Equivalence Regions

Finally, if a path is optimal with respect to equation (6.1) for two different vectors of weights $\boldsymbol{w}^i$ and $\boldsymbol{w}^j$, we call $\boldsymbol{w}^i$ and $\boldsymbol{w}^j$ *equivalent*. This implies that there exist different possible weight configurations that are indistinguishable for the user in our setting, as the corresponding paths are equal. Consequently, we call a set of weights where all elements are *equivalent* to one another an *equivalence region*. This implies that we do not need to exactly determine $\boldsymbol{w}^*$; it is sufficient to find an estimate $\hat{\boldsymbol{w}}$ that is *equivalent* to $\boldsymbol{w}^*$. The notion of equivalence weights and the resulting discretization of the weight space is key for the convergence of our algorithm, as shown in [1]. In each iteration we pick a weight from the *feasible space* that is not equivalent to a weight with a corresponding path that has been previously presented to the user. Thus, each user feedback allows us to remove at least one equivalence region from the feasible space. The algorithm terminates if all weights in the feasible space lie in the same equivalence region, i.e., all remaining feasible weights are indistinguishable to the user. As the number of paths and therewith the number of equivalence regions is finite, our algorithm finds the optimal solution in a finite number of iterations.

6.2.4 Multiple tasks

We now detail how to extend our previous work to multiple tasks. In [1], the robot was required to traverse between only one start and goal, i.e., we considered only one task at a time. We now consider a set of points of interest in the environment yielding multiple start-goal pairs. In a multitask scenario, we learn about the constraints in each interaction round by obtaining feedback for a single task. We can combine the information from

multiple rounds by intersecting the *feasible spaces* of all individual tasks. This leads to a *passive* learning effect: Obtaining feedback about a task (s_1, g_1) potentially affects the learning for another task (s_2, g_2), as some weights corresponding to paths for (s_2, g_2) might no longer lie in the *feasible* space.

Further, we discuss how to choose a new pair of paths that we present to the user. First, we consider a single start-goal pair. In our framework, we iteratively improve the robot's path. Initially, we pick a path P^0 that is optimal for $\boldsymbol{w}^{\max}$. Hence, P^0 follows the user specification, i.e., it does not violate any *avoid*- or *speed*-zones, does not traverse roads in the wrong direction and uses roads as much as possible. In each iteration we then present the user with the current best path and one alternative. If the user prefers the alternative it becomes the new current best path P^{best}. In [1] we presented two policies for finding a new alternative path, $\pi_{\texttt{vertexSearch}}$ and $\pi_{\texttt{minVertex}}$. Both perform a local search over the vertices of the current *feasible space*, similar to the pivot step in the simplex algorithm. While $\pi_{\texttt{vertexSearch}}$ starts at the weight of the current best path $\hat{\boldsymbol{w}}^{\text{best}}$, its variation $\pi_{\texttt{minVertex}}$ starts at the minimal weight within the feasible space. This more greedy approach showed a better performance in simulation; hence, we use the `minVertex` policy in the user study. The corresponding function `minVertex`$(\cdot)$ takes the following arguments: The current feasible space described by $\boldsymbol{A}$ and $\boldsymbol{b}$, the number of new weights to be returned k, the set of weights that were presented in previous iterations $\mathcal{W}_i$ and the current best estimate $\hat{\boldsymbol{w}}_i^{\text{best}}$.

In the multiple task setting we additionally have to pick a start-goal pair for which we want to present new paths. We propose a simple policy for this in Algorithm 4. Let a learning instance l_i be the collection $(s_i, g_i, \boldsymbol{w}_i^{\text{best}})$ for a task i where $\boldsymbol{w}_i^{\text{best}}$ is the weight vector corresponding to the current best path. Further, let L be the set containing all l_i for all tasks in the scenario. Given L and the current feasible space described by $\boldsymbol{A}\boldsymbol{w}^* \leq \boldsymbol{b}$, the algorithm iterates over all l_i and computes a new alternative path with the `minVertex` policy (line 4). Then, it selects the task, i.e., start-goal pair, where the time difference between the current best path and the tentative alternative is maximized (line 7). As a result the user is usually presented with those tasks for which the alternatives consist of very different paths in the first few iterations. After some user feedback is obtained, fewer paths are feasible and the respective weights are less different. Hence, in later iterations the two paths presented to the user become more similar.

In practice, the evaluation of Algorithm 4 can take significant computation time. We can approximate the selection of l_i by sampling a random subset L' of L and iterate over the elements in L' in line 2 of the algorithm.

Input: $\boldsymbol{A}$, $\boldsymbol{b}$, L
Output: $l_{\max}$
$\texttt{time\_saving} = -\infty$
$l^* = \emptyset$
for $(s_i, g_i, \boldsymbol{w}_i^{\text{best}})$ *in* L **do**
  Pick P^1 as the optimal path for $\hat{\boldsymbol{w}}_i^{\text{best}}$
  $\boldsymbol{w}_i^{\text{new}} = \texttt{minVertex}(\boldsymbol{A}, \boldsymbol{b}, 1, \mathcal{W}_i, \hat{\boldsymbol{w}}_i^{\text{best}})$
  Compute new path P^2 for $\boldsymbol{w}_i^{\text{new}}$
  if $t(P^1) - t(P^2) > \texttt{time\_saving}$ **then**
    $\texttt{time\_saving} = t(P^1) - t(P^2)$
    $l^* = l_i$
  end
end
return l^*

Algorithm 4: Choose task for learning

Impact of learning on performance

Note that the interactive learning does not guarantee improvements in the the completion time of paths. Users accept alternative paths if they have a lower cost with respect to equation (6.1), which does not necessarily imply a lower time. Consider the simple example in Figure 6.1. Following the specification leads to the direct path P^{init}, shown in purple. However, a possible alternative enters into the avoid zone (shown in red), but also traverses along a user specified road (shown in green). The alternative path (shown in yellow) might have a lower cost with respect to the user preference. This effect becomes especially relevant in the multi-task setting due to *passive learning*: Obtaining feedback for paths between some s_1 and g_1 adds inequality constraints to the *feasible space*, which then affects the optimal path between s_2 and g_2. Relating back to the example from Figure 6.1, the user might not be presented with these two paths as the learning system might infer about the importance of the avoid zone from a different task. However, in the results of the user study we analyze the performance in detail and show that in practice, the interactive learning improves the performance for most users.

6.2.5 Metrics

To quantitatively measure the quality of specifications, two metrics are employed: *Entropy Ratio* and *Time Ratio*. Similar metrics were originally introduced and applied to user

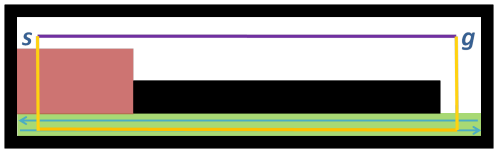


Figure 6.1: Example for increase in task completion time. The initial specification results in path P^{init}, shown in purple. An alternative solution (yellow) might have a longer traversal time, but correspond better to the user preferences if they value the avoid zone as less important and prefer the use of the road.

specifications as part of our earlier work in [8].

Entropy Ratio

Given a graph G and a specification Γ, the entropy quantifies the complexity of the robot's action space, generated by the combination of the environment and user specification, by considering the number of outgoing edges available at each node, taking their cost into account. The entropy ratio is expressed as the ratio of entropies between graph G^{Γ} and graph G (See Section 6.2.2), i.e., the constrained and unconstrained environment. We measure complexity using entropy, defined similarly to previous work in [8] and [30]. Given an estimate $\hat{\boldsymbol{w}}$ of the user weights, let the cost of an edge be the sum of time and the estimated weight: $\hat{c}(e) = t(e) + \hat{w}(e)$. Further, let $\hat{c}^{\min}(v_i, v_j)$ be the minimal cost between all parallel edges from v_i to v_j.

For a given vertex v_i on a graph and the set of its neighbours $\mathcal{N}(v_i)$, the entropy of v_i is given by

$$H(v_i) = -\sum_{v_j \in \mathcal{N}(v_i)} p(v_i, v_j) \log_2 p(v_i, v_j), \tag{6.4}$$

where we define $p(v_i, v_j)$ as

$$p(v_i, v_j) = \frac{\frac{1}{\hat{c}^{\min}(v_i, v_j)}}{\sum_{k, v_k \in \mathcal{N}(v_i)} \frac{1}{\hat{c}^{\min}(v_i, v_k)}}. \tag{6.5}$$

To obtain the entropy of a graph, we take the sum over the individual vertex entropies:

$$H_G = \sum_{v_i \in V} H(v_i), \tag{6.6}$$

where V is the set of vertices of a graph G, and H_G is the entropy of a graph. The entropy ratio is then denoted by $\eta = {}^{H_{G^\Gamma}}/_{H_G}$. The entropy is maximized for H_G, when there are no user specifications the robot can move freely in any obstacle-free regions of the environment. Adding constraints always decreases entropy as the robot's movement becomes more restricted. Thus, large entropy ratios indicate rigorous specifications where the robot behaves in a more predictable way.

Time Ratio

Given a graph G, a specification Γ and a set of start and goal pairs V' where each start and goal is a vertex on G, the *time ratio* metric describes the effect of the constraints Γ on the average duration of the shortest paths with respect to equation (6.1), i.e., the ratio between the average optimal path durations in graph G^Γ and in graph G. Thereby, paths for all pairs in V' are considered. Similarly to our previous work [8], we distinguish two forms of the metric: The *global* time ratio considers all vertices on the graph, i.e., $V' = V \times V$ where V is the set of vertices on G, while the *task* time ratio considers only a defined set of start and goal pairs, i.e., $V' = \{(s_1, g_1), (s_2, g_2), \dots\}$.

6.3 Study Design

6.3.1 Scenario Description

The study scenario describes a simulated industrial environment adapted from the layout of a real world facility. An autonomous mobile robot is required to fulfill material transport tasks; a single task consists of navigating from a given start to a given goal location.

Users are provided with a description of the environment detailing different zones as illustrated in Figure 6.2. This includes areas with high pedestrian traffic, loading docks, storage space, robot parking and charging, dedicated human work and break areas and, an assembly line. The tasks consist of navigating between the labeled areas, for instance traversing from the robot charging zone to the upper end of the assembly line. The user is asked to generate a specification such that the robots are able to reach any of the areas, excluding the human break rooms. Further, robots are never allowed to cross through the assembly line.

Before starting the specification task, users receive an explanation of the traffic rules and instructions on how to use the interface (For more details see Section 6.3.2). One-way

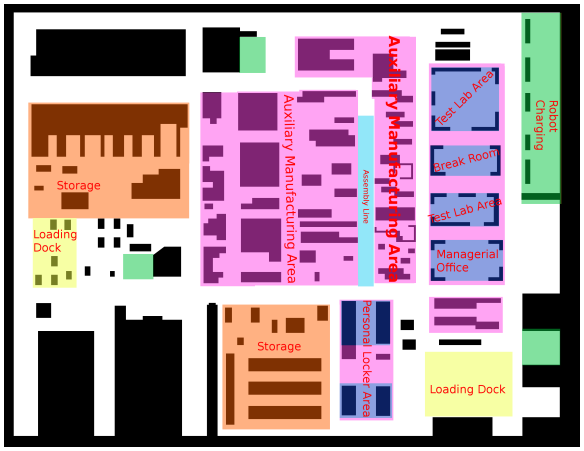


Figure 6.2: The scenario described in the study. Black corresponds to physical obstacles while white is the free space. The described areas in the environment are labeled as follows: High pedestrian traffic – purple, loading docks – yellow, storage – orange, robot parking and charging –green, human work and break areas – dark blue, assembly line – light blue.

roads are described as encouraging the robot to traverse them in the direction of traffic and discouraging the robot from traversing in the opposite direction. Two-way roads function as two adjacent and opposing one-way roads. Areas of avoidance simply define a part of the environment where robot traffic is undesired, while speed-limits express that the robot is required to drive with a reduced speed in a specific area.

Finally, users are told that "the robot can navigate freely in the environment without any traffic rules" and has fundamental safety features such as obstacle avoidance. The traffic rules are "meant to guide the robot's behaviour" in a way preferable for the user, and users are free to specify as few or many rules as they deem necessary.

6.3.2 Study Procedure

Structure

The study was approved by the Office of Research Ethics at the University of Waterloo (ORE #31689). Each study session took approximately 1 hour. The study process is structured in three parts: overview and introduction, training and main study. In the first part, the scenario and the role of the participant are briefly explained. A video introduces the user interface and demonstrates how to create traffic rules in detail. Further, written

information about the traffic rules and the robot's capabilities is provided.

In the training phase the objective is to familiarize participants with the interface. They are presented with a smaller example environment including a similar description as in Figure 6.2 and are asked to create traffic rules until they feel confident in using the interface. At the end of the training, participants tele-operate the robot in the simulated environment.

The main study has three phases: specification, interaction and tele-operation. The first two phases are illustrated in Figure 6.3. In the specification phase, participants are presented with the environment from Figure 6.2 and the written instructions on the traffic rules and the robot's capabilities. It is once more stated that the robot is required to navigate between all marked areas on the environment (with the exception of the dedicated human break areas) and that the robot is able to navigate without any traffic rules present. Participants are then asked to define the traffic rules they find appropriate to achieve the desired robot behaviour. Once they are satisfied with their set of traffic rules, the first phase is concluded.

In the interaction phase, users are iteratively presented with two alternative paths for a task. First, a brief instruction explains the interface: On the map of the environment both paths are shown simultaneously, a simple menu allows participants to select a path, which is then highlighted in color. Further, if a path is highlighted and violates a *penalty* constraint, the perimeter of the constraint in question is also highlighted. Finally, the menu features information about the duration of the two paths and lists the violated penalty constraints. All participants go through 20 iterations of the interaction process, unless the learning algorithm cannot find a new path for any of the tasks and terminates earlier. The task for which they are presented with two alternatives is selected by Algorithm 4 using an approximated set L' containing five tasks, sampled randomly.

In the last phase of the study, participants are asked to tele-operate the robot from a given start to a goal location. Thereby, they can choose freely if they follow their own traffic rules or violate them.

Questionnaires

During the study, participants are also presented with several questionnaires. Before providing the specifications, users indicate their trust in the robot's capabilities to fulfill the described tasks. After each of the main steps users are presented with a questionnaire where they rate how well they specified the traffic rules and how confident they feel about

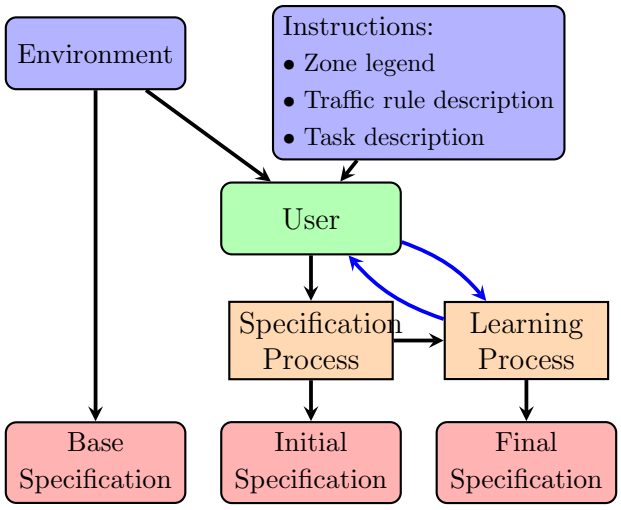


Figure 6.3: Flowchart of the study with the resulting specifications, black arrows are only executed once while blue arrows are executed multiple times. Participants initially receive an instruction set and a description of the environment. The environment yields a base specification, only including obstacles. Using the traffic rules they create the initial user specification for the robot. During the learning interaction users provide feedback, leading to a revised, final specification.

using the system. Further, during each step of the interaction participants are asked how acceptable both paths were and, in every third iteration, what their reasoning for their choice was. Finally, the study concluded with a longer questionnaire where users evaluate the overall system. We use the standardized system usability score (SUS)[78] for evaluating the interface design and additional questions focusing on the warehouse scenario.

Evolution of specifications

We define the specifications corresponding to different stages of the process, detailed in Figure 6.3. Before a user specification is provided, the environment including the obstacles yields a base specification where the optimal paths P_i^{base} for each task only minimize time. After receiving the traffic rules but before the learning we obtain the initial specification. The optimal paths P_i^{init} are the optimal paths corresponding to $\boldsymbol{w}^{\max}$, i.e., the paths that categorically follow the initial specification if possible. After the learning process, a unique solution is not necessarily obtained as we are not guaranteed to achieve convergence to the optimal weight in the given number of iterations. Hence, we need to pick some $\boldsymbol{w}^{\text{final}}$ from the resulting *feasible space* $\mathcal{F}_{\text{final}}$ after learning (See Section 6.2.3). We propose a conservative approach for determining the final specification by choosing P_i^{final} to be the optimal paths for the maximum feasible weight, i.e., $\boldsymbol{w}^{\text{final}} = \underset{\boldsymbol{w} \in \mathcal{F}_{\text{final}}}{\arg\max}\{\mathbf{1} \cdot \boldsymbol{w}\}$.

6.3.3 Hypotheses

Finally, we propose the two main hypotheses for the user study.

Hypothesis 2 (H2) *The learning process has the following properties: (a) user accept alternative paths that violate some of the constraints they specified over the course of the learning process and (b) the task performance improves through the interaction process.*

Hypothesis 3 (H3) *Users find the specification process, and the interaction with the learning system intuitive and efficient.*

Hypothesis 2 focuses on quantitative analysis of the user interaction. For (a) we analyze the user feedback from the interaction while (b) is based on the metrics described in Section 6.2.5. To validate Hypothesis 3 we conduct quantitative analyses based on the questionnaire, including the SUS score, as well as a qualitative analysis using the free form user comments.

6.4 Results

6.4.1 Participants

For the study we recruited 43 participants (30 male and 13 female) via mailing lists. In total 24 participants are affiliated with the Faculty of Engineering at the University of Waterloo. Moreover, 28 participants are pursuing or have completed a graduate education while 14 are pursuing or have completed an undergraduate degree. Finally, 12 participants stated that they have background knowledge in robot motion or urban planning.

The population of the study consists of four groups: 21 novice users and 10 repeat users, 9 researchers from the NSERC Canadian Robotics network (NCRN), and 3 Clearpath employees. The novice users had never interacted with the presented framework before, while the repeat users had previously used the system once (e.g., during the pilot phase of the study). The NCRN participants volunteered to take part in the study during the 2019 NCRN Trials, and the Clearpath employees volunteered to take part in the study during a corporate hackday. No participant is part of more than one group. In Sections 6.4.2–6.4.4 we present results for all users while Section 6.4.5 focuses on differences between the different groups.

6.4.2 Specifications

The initial specifications provided by the users vary in their complexity. We summarize the characteristics of the initial specifications in Table 6.1. Recall that the number of user defined roads does not correspond to the number of constraints for the planning problem as roads constitute a reward and a penalty constraint for each lane. We show three example specifications, the smallest and largest with respect to the number of traffic rules as well as the specification from participant 5 that is illustrated in Figure 6.4.

6.4.3 Hypothesis 2

(a) Acceptance of alternative paths For each user we define $\alpha^j_{\texttt{all}}$ to be the percent of iterations in which user j accepted the alternative path. Further, we introduce $\alpha^j_{\texttt{tasks}}$ as the percentage of the tasks presented to the user where user j accepted at least one alternative, i.e., where they rejected the initial path at some point.

Overall 41 out of the 43 participants accepted at least one alternative path. On average we found that $\alpha_{\texttt{all}} = 0.42$, meaning that users accepted alternatives in 42% of the

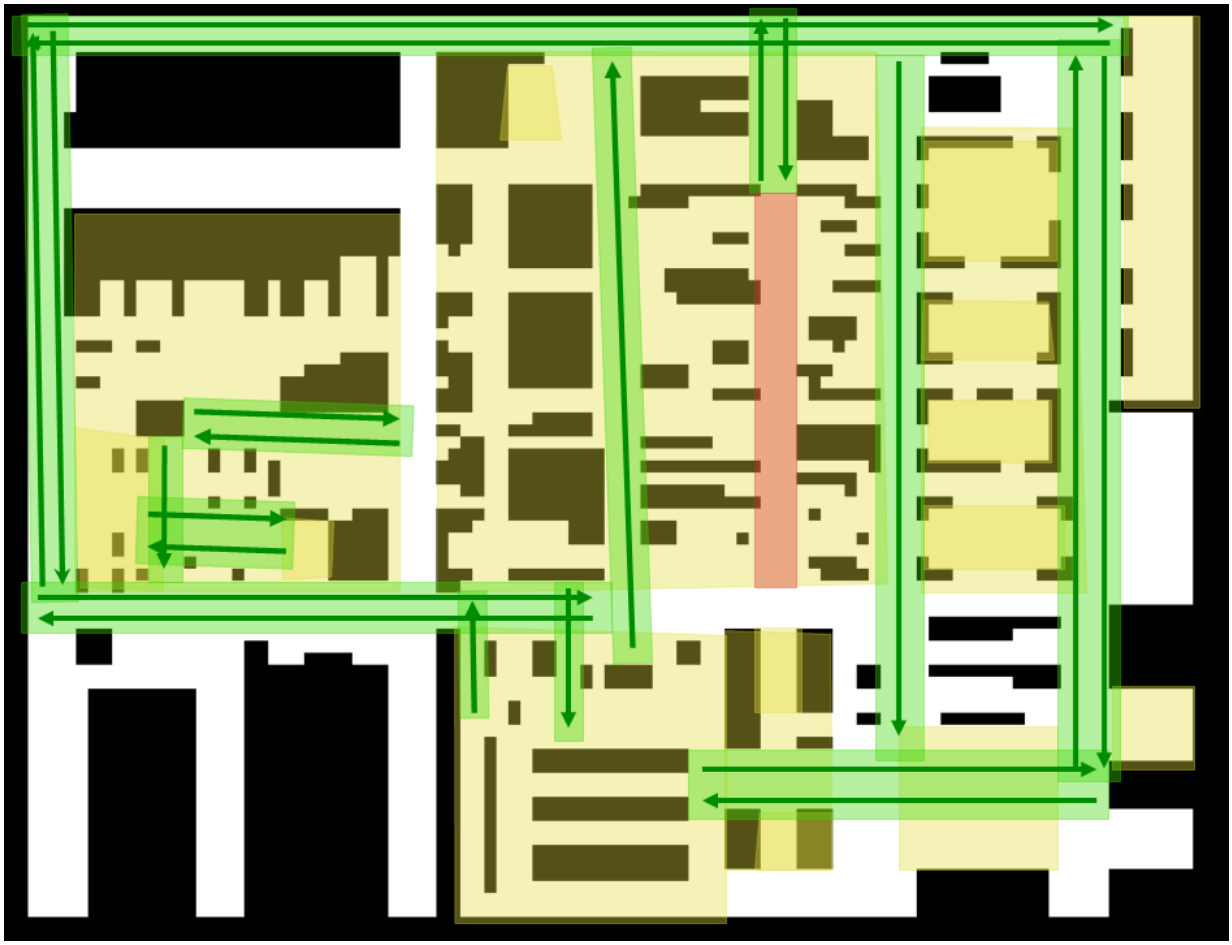


Figure 6.4: Example specification from participant 5. Reduced speed rules are marked in yellow, road rules in green, and avoidance rules in red.

Table 6.1: Characteristics of the initial user specifications. We show the individual mean, median, min and max for each type of constraint and the number of traffic rules. Further we report the characteristics of the overall smallest and largest specification as well as the example of participant 5 (P 05), shown in Figure 6.4.

	Roads	Avoidance	Speed	Traffic Rules
mean	11	4	8	23
median	9	4	7	24
[min, max]	[0, 24]	[1, 12]	[0, 19]	[10, 40]
EXAMPLES				
smallest	4	1	5	10
largest	22	1	17	40
P 05	13	1	17	31

interactions. The task related acceptance has a mean of $\alpha_{\texttt{tasks}} = 0.58$; thus, for over 1 out of 2 tasks users preferred an alternative path over the initial one.

Further, we investigate the correlation of $\alpha_{\texttt{tasks}}$ and the richness of user specifications. We characterize the richness of a specification in two ways: The number of traffic rules that the user defined and the number of resulting constraints for the planner. We found no correlation between the number of traffic rules and the acceptance rate, and only a weak, but statistically significant Spearman rank correlation between the number of constraints and the acceptance rate ($\rho = 0.37$, $p < 0.05$). This indicates that users who define a larger set of traffic rules are not necessarily more likely to accept alternative paths, unless the traffic rules created are themselves generally more complex (i.e. multiple constraints needed to encode each rule, e.g. roads in our scenario).

Together with the task specific acceptance rate of $\alpha_{\texttt{tasks}} = 0.58$ we find strong support for our first hypothesis: Users are unaware of the impact of their specification and thus allow robots to violate traffic rules (or use roads less frequently) when presented with different possible solutions. Further, the obtained revised specification leads to paths where users chose an alternative path over the initial one in 42% of cases. Moreover, the correlation of complexity and acceptance shows that this effect becomes more apparent for users defining many complex traffic rules that translate to multiple constraints each. In [1], Wilde et al. postulated three types of users for the simulations: a low trust user with many constraints for which the importance varies drastically, a high trust user with few constraints that all are relatively important and an intermediate user. In the current user study we do not observe a discrete separation but rather a continuous distribution for the user behavior.

From the difference in the correlation we can conclude that users defining many road rules are more likely to accept deviations from the initial path.

(b) Increased performance To evaluate the changes in the performance, we compare the *time ratio* metric of the initial and the final specification, illustrated with violin plots in Figure 6.5. Further, we compare the metric for global and task-dependent evaluation.

For both evaluations we observe a decrease in the time ratio after the learning process as well as a reduction in the standard deviation. A paired-samples one-sided t-test was conducted on the task time ratio between initial and final specifications. The task time ratio of initial specifications ($M = 1.48,\ SD = 0.32$) was found to be significantly different ($p < 0.01$) from that of final specifications ($M = 1.28,\ SD = 0.19$).

Unsurprisingly, the initial specifications vary largely in their impact on the performance as the number of traffic rules users defined range from 10 to 40. However, the decrease in the population standard deviation following interaction indicates that the learning reduces the variation in the performance impact of user input and thus helps users to create more efficient task specifications.

Further, we notice that the task-dependent time ratios are higher than the global ones for the initial and final specifications. The global metric takes into account locations that are less relevant in the scenario, e.g., the lower left corner of the environment (shown in Figure 6.4) is not part of a robot task and therefore neglected by most users. Moreover, as the global evaluation considers all vertices on the graph, many close-by pairs of vertices are considered, where the specification often has little influence. While the task specific performance is worse, the relative change in time ratio is higher in the task specific case (19.5%) compared to the global metric (16.7%). Hence, the learning effectively improves the performance of the tasks in the scenario.

Figure 6.6 illustrates the change of the task-dependent time ratio and the entropy ratio for all user specifications. From the plot, we observe that while the time ratio decreases the entropy ratio increases. Entropy corresponds to how predictably the robot behaves. It captures the robot's degree of freedom with respect to the edge cost on G^{Γ}, which is the sum of time and user weight. As the learning process initially assumes high weights on the constraints and thus only reduces weights after obtaining feedback, the entropy increases. After learning, the robot might be allowed to violate some constraints, which enables more options to navigate in the environment. This leads to fewer restrictions on robot behavior, which may not always be desirable. However, this relaxation of the specification is traded-off with the increase in performance.

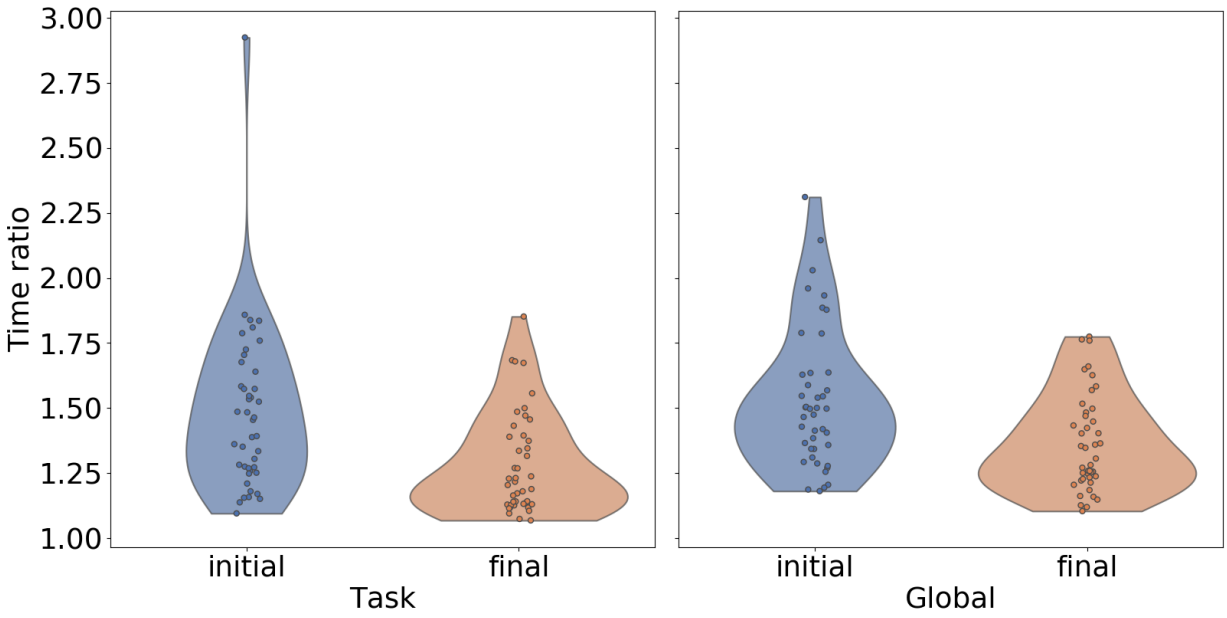


Figure 6.5: Change in the task-dependent and global time ratio metric of the specification due to active learning. In both plots the left bar shows the time ratio of the initial specification, averaged over all users. The right bar illustrates the time ratio of the final specification, also averaged over all users.

Running a paired-samples one-sided t-test, we found the entropy ratio of initial specifications ($M = 0.879,\ SD = 0.063$) to be significantly different ($p < 0.01$) from the entropy ratio of final specifications ($M = 0.910,\ SD = 0.048$). Moreover, we notice that while the mean entropy increases, the standard deviations of the time and entropy ratios decrease due to the learning. With respect to the metrics, the specifications become more similar during the learning. Two sample f-test between the initial and final time ratios show a significant difference in the variance of the task and global versions of the metric ($p < 0.01$ and $p < 0.05$ respectively). No significant difference in the variances was found when performing the two sample f-test between the initial and final entropy ratios of the specifications. Additionally, Figure 6.6 also suggests that the specifications with low initial values of entropy and high initial task time ratios generally see more improvement following the preference learning process. This is verified by a strong Pearson correlation between the initial values and the difference between the final and initial values, resulting in $\rho = -0.80$ ($p < 0.01$) for time ratio, and $\rho = -0.77$ ($p < 0.01$) in the case of entropy ratio.

In summary, the learning system leads to a significant improvement in the time ratio metric, especially when measured for the tasks in the scenario. Further, the learning revision reduces the variance in performance (time-ratio) between different specification, an effect that is more pronounced with the task time-raito. Moreover, specifications that are initially more inefficient benefit more from the learning process.

6.4.4 Hypothesis 3

We now report on the users' assessment of the usability of our framework, based on the system usability score (SUS). While the SUS does not provide a grade for the usability itself, the work of [78] provides a reference frame based on $2,324$ surveys using the SUS. In our study, users gave a mean SUS score of 69 while the median is 72.5. The difference arises from two outliers in the data set with a difference from the mean of over 2 and over 3 standard deviations. The mean corresponds to the second highest quarter of all surveys examined in [78]. Specifically for computer based GUIs, [78] reports a mean of SUS of 75.

After the three main parts of the study – constraint specification, learning interaction and tele-operation – participants were asked to asses how well they specified the robot behaviour on a 1 to 10 scale. On average, users reported similar ratings at each step, varying between 7.0 and 7.8, with standard deviations between 1.6 and 1.7. Hence, users felt relatively confident about how they used the framework. Interestingly, we observe an inverse correlation (Spearman coefficient -0.65, $p < 0.01$) between the second self assessment and the richness of the specification, i.e., the number of constraints. Users

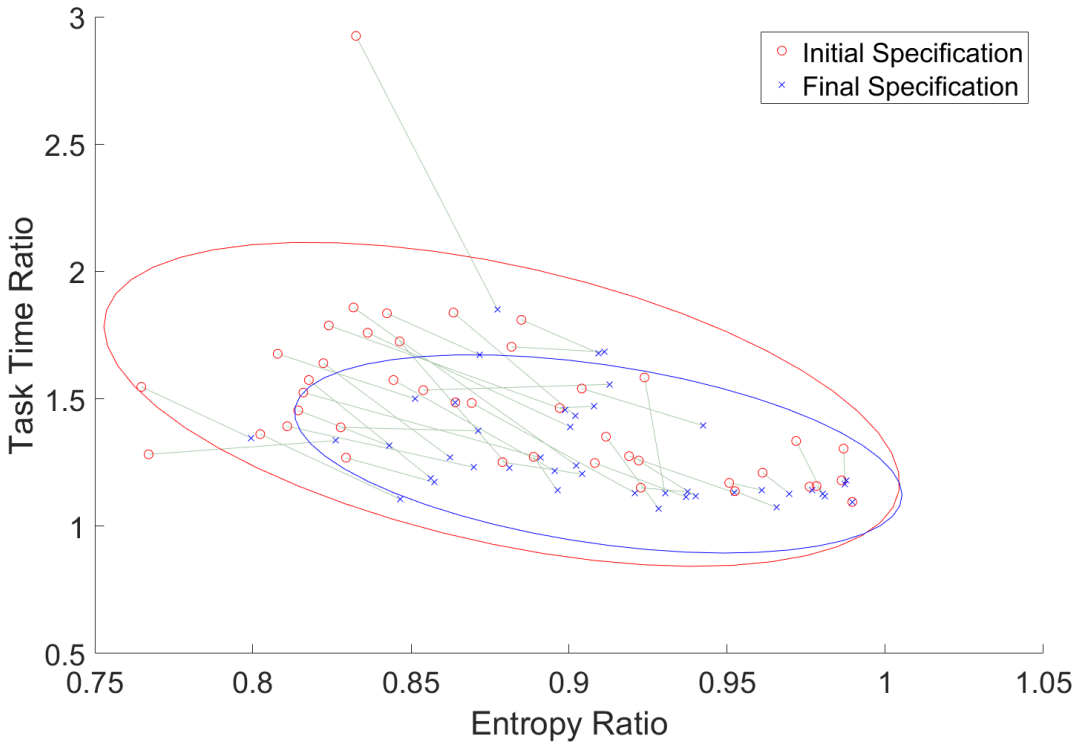


Figure 6.6: Change in the task time ratio and entropy ratio metrics of the specifications due to active learning. Red indicates the metrics for the initial specification, blue shows the metrics for the final one, and the lines associate the initial and final specifications of each individual users. The elipses represent the 95% confidence intervals.

defining a larger set of constraints tended to view their specification more critically after the learning. Thus, the interactive framework helps users to better understand the impact of their specification on the robot's performance.

6.4.5 Differences in the population

Acceptance rates

When splitting the data into the repeat, novice, and NCRN populations, we observe only a minor increase in the acceptance rates for the novice users compared to the repeat ones, but a large (yet not statistically significant) decrease in the acceptance rate of NCRN participants compared to the repeat ones. However, the correlation of acceptance rate and complexity of the specifications disappears for the repeat and NCRN users while it is stronger for novices. Repeat users are more aware of the impact of their specifications while novice user benefit from the interaction to improve the robot's behaviour.

The size of the Clearpath employee population is too small to assess quantitatively. However, it is notable that one of the Clearpath employees did not accept any of the proposed alternatives, and thus required the robot to strictly follow their specification. This behaviour was observed in only one other participant from the other population groups.

Time ratio metric

Between novice and repeat users the *time ratio* metric varies. We recall that the task-dependent metric better reflects the effect of specifications on the task performance, and thus we show the results for the task-dependent time ratio in Figure 6.7. We observe that the initial specifications provided by the novice users show a larger variance compared to repeat users. While the median values are relatively similar, the distribution for novice specifications spreads out to higher time ratios. However, the time ratios of final specifications are much more similar.

A two-sample one-tailed t-test was performed on the difference in time ratio between the initial and final specifications of novice and repeat users, which revealed that the two populations are significantly different ($p < 0.01$). In other words, the changes in the time ratios differ between the two groups.

Analyzing the NCRN participant data, the initial task time ratio of these participants is not significantly different from either the novice or repeat users. Despite this, we find

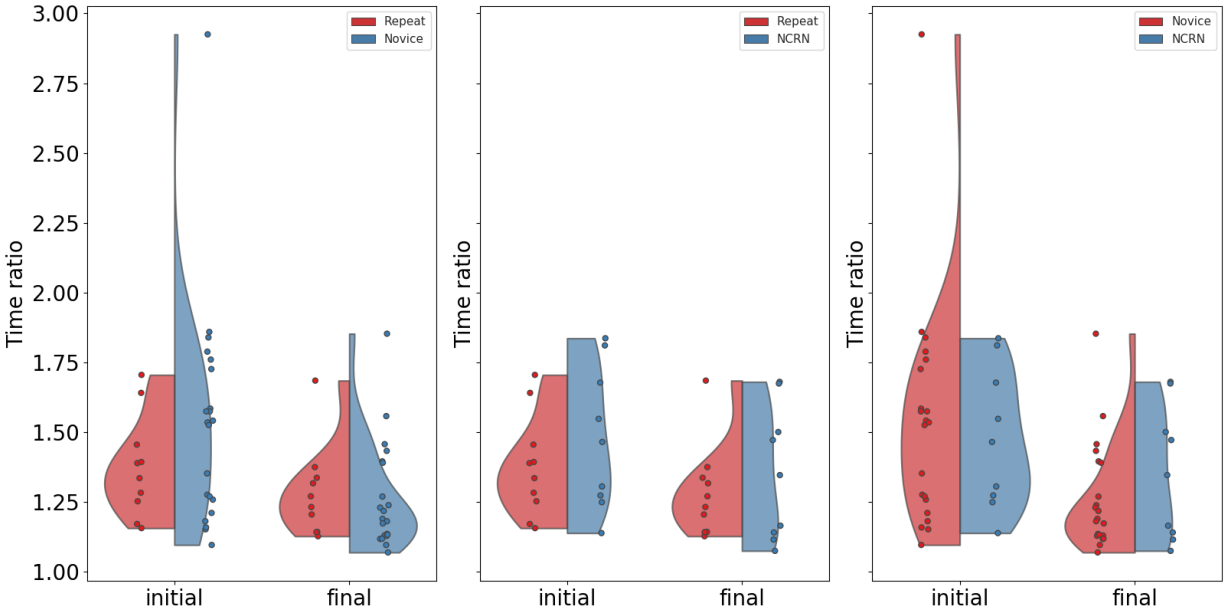


Figure 6.7: Change in the task-dependent time ratios of the specification, comparing novice, repeat, and NCRN users pairwise.

a statistically significant difference ($p < 0.05$) between the the difference in time ratio between the final and initial specifications of NCRN participants, and that of novice users. No such statistically significant difference was observed in the time ratio improvement of NCRN users when compared to repeat users. These results show that the NCRN participants are generally between novices and repeats, although their improvement is a lot more similar to that of repeat users.

We conclude that novice users create more diverse specifications with respect to the impact on performance. However, the learning process helps them to improve the specification and obtain better specifications. Repeat users seem to have a better understanding of the effect of the traffic rules and thus design specifications more carefully. Consequently, they allow for fewer violations that effectively render constraints insignificant and therefore obtain a smaller time benefit. NCRN participants allow even fewer violations still. Despite this however, their performance scores place them solidly between repeat and novice users.

Entropy ratio

Although small differences in the mean entropy ratios across the two populations were observed for both the initial (0.890 for novice, 0.870 for repeat, and 0.881 for NCRN) and final (0.922 for novice, 0.901 for repeat, 0.903 for NCRN) specifications, these differences were not found to be statistically significant.

SUS score

The mean SUS of repeat users is 71.8 (median 76), while the mean of novice users is 69 (median 75), and the mean of NCRN users is 64 (median 65). Naturally, participants who have interacted with the UI before are likely to find it more easy to use. Nonetheless, the reported difference is less than half of the standard deviation among all users scores and thus is not statistically significant. While not significantly so, on average, the NCRN participants rated the interface with the lowest SUS rating. One reason for this could be that the NCRN participants took part in the study while in a lab room, surrounded by other researchers and experiments. This was in contrast to the novice and repeat users who took part in the study in much quieter and less distracting environments. Anecdotally, the NCRN participants were observed to be less singularly focused, and more often interrupted by the events around them.

6.5 Discussion

6.5.1 User feedback

While most users ranked the interaction with our system as positive, participants provided several suggestions for improving the framework in the questionnaire feedback. Almost half the users expressed the desire to change their specification during the learning process. This indicates that although instructed on general robot behaviour, participants found it somewhat difficult to envision how all of the created traffic rules affect the behaviour of the robot. They could be well served by visualizations showing robot behaviour during the specification phase.

Another aspect that could be improved is how the user feedback is incorporated into the learning. Currently users express their traffic rules preferences is by selecting the preferred path. While this approach is intuitive and simple to use, users occasionally expressed

frustration when both paths presented to a user contain undesirable behaviour, and so users have to select the lesser of two evils. As a result, future work should investigate additional forms of feedback that might better reflect a user's preference, and could potentially lead to a more efficient learning process.

The work of [64] investigates richer forms of feedback in active preference learning. In addition to ask for the user's preference, feature queries give the user the opportunity to express their reasoning, i.e., "Which feature is most responsible for the difference in your preference between these two trajectories?". This aligns with feedback from the questionnaire: Some participants stated that they rejected alternative paths as they violated both a minor and a major constraint at the same time; the violation of only the minor constraint would have been acceptable, but that was unknown to the learning system. In this case richer user feedback would help in two ways, allowing users to express the reasoning for their path selection, and potentially reducing the number of iterations of the learning. On the other hand, a drawback of this approach is the increased complexity of the interaction.

Another approach for richer feedback could allow users to manually indicate, and potentially correct, the undesired sections of presented paths. This idea is investigated by [79] where users segment a robots trajectory into good and bad parts.

Based on the suggestions of one of the Clearpath participants, we could also attempt to improve the starting point of the iterative process. The suggested method involves two modifications to the interaction process: allowing participants to specify the importance of a constraint at specification creation time, and showing participants the learned importance weights at the end of the iterative process. This way, with successive deployments, a user might become more aware of their own preferences, and the weights that would result in more appropriate behaviour. The user could then use this information to seed future deployments with initial weights, achieving better robot behaviour in initial specifications, and shortening the preference learning process by requiring fewer iterations. The effectiveness of such a system remains to be investigated, as it is unclear that copying a set of constraints (together with their learned weights) to a new environment would result in similar patterns of robot behaviour.

A participant also noted that a user's acceptance of violations could vary depending on the urgency of the overall robot task, which is something that should be communicated to the user. So if the robot is tasked with completing as many tasks as possible in a unit of time, an operator of the robot might be very accepting of violations. On the other hand, if the robot is not time-pressured in any way, an operator might prefer for the robot to take its time and follow the specification more strictly.

6.5.2 Repeat and novel users

In Section 6.4.5 we have shown that specifications originating from novice, repeat, and industrial robotics expert users differ in the time ratio metric. This indicates that there are differences in how the groups of users specify the robot constraints, and that these metrics could be used to identify the expertise of a user based on the specification provided. In multi-user systems, this could be used to combine multiple specifications, emphasizing those of expert users. Despite the observed differences, the iterative preference learning system was shown to be capable of improving the specification performance of all types of users. This leads us to hypothesize that even in the case of specifications designed by domain experts, the learning framework could still be used to help increase specification performance.

6.5.3 Learning framework

Previously, in [1], the active learning framework was evaluated in simulation. Validating the extended algorithm proposed here in the user study allows us to make additional observations about the practicality of the approach. Unlike the work of [55, 54, 61, 62], the learning framework used in this study is currently based on a deterministic user model. The major drawback is that our model does not consider users who behave differently than described in the assumed cost function. Nonetheless, we were able to demonstrate that using a simplified linear user model, the framework proposes alternative paths that users accept over the initial paths and revises the specification to improve the task performance within a small number of iterations. While the resulting final specification does not necessarily correspond to the optimal solution with respect to the hidden user preferences, the deterministic model allows for quick learning, yielding substantial improvements within only 20 iterations. A more complex, potentially probabilistic user model would make fewer assumptions about the user's behaviour and thus be more robust; however, usually at the cost of performance, i.e., the number of iterations required for learning in a comparable setting.

Further, due to the multitask scenario we were able to observe some inaccuracies in the user feedback with respect to our user model. When learning about a single task, the *feasible space* can never be empty as the algorithm stops when all feasible weights are equivalent. However, intersecting the feasible spaces for different tasks can lead to an empty set. In that case, the user feedback to different tasks contradicts one another, assuming the linear cost function. Notice that an empty intersection of the feasible spaces is not a necessary but a sufficient condition for inaccurate user feedback.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we proposed a method to assess the quality of user specifications describing the behaviour of material transport warehouse robots. The proposed method provides a set of measurements that captures the positive and negative effects of a robot's behavioural constraints. These metrics measure the behavioural predictability of the robot, the extent to which robot behaviour meets a user's expectations, and the loss of efficiency due to following the user specifications. A series of two user studies were designed to demonstrate and validate these metrics.

The first was a preliminary user study, where novice users generated specifications for a set of warehouse robot tasks taking place in a simulated environment. Our metrics illustrate that users given the same task description generate very different specifications, and that differences between specifications have significant impact on robot predictability, legibility and efficiency, which may not be readily apparent to novice users. This was best exemplified by a participant's faulty specification that did not allow the robot to actually complete the task. The participant was unaware that the robot was unable to complete its task. Additionally, users were not necessarily capable of accurately assessing their performance in creating these specifications. This motivated the need for a system that is able to improve user specifications, especially those originating from novice users.

In the second part of the thesis, the specification system was integrated with a preference learning algorithm [1], and our metrics were successfully applied alongside the active learning framework, which helped users create better specifications. The study showed

that the algorithm allows for a substantial improvement of specifications while requiring few iterations of user interaction. We were able to use the metrics to show that the algorithm significantly improved the performance of specifications for a variety of users. In particular, users who generate initial specifications most detrimental to robot performance received the most benefit from the interaction, resulting in final specifications with similar performance across users, and thus reducing the need for user training.

7.2 Limitations Of The Work

The main limitations are related to the lack of domain expert users as participants in our studies, and the inability of our metrics to accurately account for dynamic environments and certain behavioural constraints.

7.2.1 Domain Expert Users

A major limitation of the results presented in this thesis is that they were obtained from participants who are not domain experts in warehouse/industrial logistics or warehouse/industrial robotics. A key aspect of our research is understanding how users program the behaviour of a robot in the warehouse/industrial space, and while we have tried to get our participants in the right mindset through the design of the scenario presented in the study, we cannot be sure that our findings extend to the target users. For this reason, we attempted to recruit expert users to take part in our studies, but were only able to recruit 3 due to the challenge of recruiting these types of participants.

7.2.2 Metrics

With different behavioural constraints, the quality of robot behaviour could be described by more than just how long trips take, and how constrianed the robot is in the environment (i.e. our proposed metrics). Through the learning process, we currently find that violations are accepted or not based on the time savings they provide. However, certain constraints could allow for differentiating between deep and light violations. For example, a user could indicate that it is preferable for the robot to skim an Avoidance zone for a longer time, rather than cut right through it for a short amount of time. These cases are not supported by our proposed metrics, as they don't capture the extent to which the robot violates a constraint

Our system also assumes a static and known environment. As such, obtaining accurate and representative metrics for a highly dynamic and changing environment is more difficult. In these scenarios the task of specifying robot behaviour can be a continuous process, and a performant and well-behaving specification could easily become suboptimal. Some of these environment changes may not be accurately captured by our metrics due to the underlying deterministic world model used in assessing the specification.

7.3 Future Work

In the future, we plan to expand recruitment of participants that are domain experts in the warehouse/industrial fields; apply revisions to the specification while the learning process has started, without having to delete all the learnt progress; and investigate other ways to provide feedback to the learning process.

7.3.1 Domain Expert Users

As previously discussed, one of the main limitations of our work is the lack of domain experts in our participant pool. Conducting our experiments with participants of the right background would be especially useful in showing the capabilities, and finding the limitations of our system. There are several challenges in recruiting domain experts to be participants in our study, including scheduling, and the workers' availability to take part in our study when it's not part of their job description. To lessen the time/effort required to take part in our study, we have begun work on making our system web-accessible, which includes fully re-implementing our interface as a Web application. We believe that this will greatly reduce the friction involved in taking part in our study, which will help us recruit representative participants.

7.3.2 Specification Revision

A common feedback from participants was the request to modify their specification after the learning process has started. After seeing how the robot behaves when following their specification, participants would identify areas of the map where they knew they could improve their specification. Instead of being able to pause the learning process, make the desired modifications, and then resume it, they were at times forced to inefficiently guide the robot's behaviour through the iterative choices provided.

One way to greatly reduce this problem is by providing feedback regarding robot behaviour while the participants are creating their specifications. This is relatively easy for scenarios that assign a small number of tasks to the robot, but effectively displaying robot behaviour in the case of many potential tasks becomes quite difficult. However, even with a well implemented display, there could still be cases where users might wish to change their specifications, especially once the preference weights start getting adjusted.

Allowing for the general modification of a specification during the learning process will also open the door for other extensions to the system, increasing its usability and effectiveness. For example, the learning system could realize that the user has different preferences for robot behaviour in a constraint, depending on where in the constraint the robot is, e.g. skirting just inside the edge of an Avoidance zone could be considered acceptable by a user, while they might feel much more strongly about the robot navigating deeply into the Avoidance zone. If the system were able to break the single Avoidance zone into two (e.g. thin outer ring, and the rest), then it would be able to effectively learn the user's preference. The learning system could also be designed to detect inefficient areas in the environment (i.e. high time ratio for paths originating or arriving in that location) and ask the user if any behaviour constraints should be created to lessen the existing negative impact.

7.3.3 Richer Feedback

Another prevalent comment received from our participants indicates their wish to provide better feedback to the learning system. This comment would usually arise from situations where both paths presented in an iteration were equally preferred, and yet we would ask them to pick one or the other. Another situation that would elicit this comment was the participant being pleased by the majority of one of the paths, but very displeased by some other parts of it. In most cases this would force the participant to pick the alternative path, even though they preferred the good portions of the original path.

There are a variety of alternative feedback interactions, of various complexities, that could be used instead or in addition to the existing one. One possible option could involve asking the users to rank each of the paths shown, and use that information as part of the learning process, expanding the area and constraints that the system is learning about from that one iteration. Alternatively, we could also allow participants to demarcate any problem sections in the paths shown. Currently, if a path violates multiple constraints and is rejected, it is not clear which of the violations caused the rejection. In addition to marking problem areas of each path, the system could also allow users to correct the paths, indicating what kind of behaviour the robot should display.

Richer forms of feedback could also aid with interacting with different types of users in a more efficient manner. Whereas now the only interaction is with a supervisor/operator user, additional forms of feedback could be created to be used with other types of users. For example, a floor worker, operating as a peer, could indicate to the robot that it should avoid a hallway due to some hazard beyond the sensor range of the robot. However, since this behaviour preference came from a peer, the system could decide to place a lesser weight on the feedback, or consider the feedback, but have it expire after a certain amount of time.

References

[1] N. Wilde, D. Kulić, and S. L. Smith. Learning user preferences in robot motion planning through interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 619–626, May 2018.

[2] Javier Minguez, Florant Lamiraux, and Jean-Paul Laumond. Motion Planning and Obstacle Avoidance. In *Springer Handbook of Robotics*, pages 1177–1202. Springer, Cham, 2016.

[3] Munjal Desai, Kristen Stubbs, Aaron Steinfeld, and Holly Yanco. Creating Trustworthy Robots: Lessons and Inspirations from Automated Systems. *Proceedings of the AISB Convention: New Frontiers in Human-Robot Interaction*, April 2009.

[4] Bonnie M. Muir and Neville Moray. Trust in automation. Part II. Experimental studies of trust and human intervention in a process control simulation. *Ergonomics*, 39(3):429–460, March 1996.

[5] Raja Parasuraman and Victor Riley. Humans and Automation: Use, Misuse, Disuse, Abuse. *Human Factors*, 39(2):230–253, June 1997.

[6] M. A. Goodrich and D. R. Olsen. Seven principles of efficient human robot interaction. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3942–3948, October 2003.

[7] Raja Parasuraman, Thomas B. Sheridan, and Christopher D. Wickens. Situation Awareness, Mental Workload, and Trust in Automation: Viable, Empirically Supported Cognitive Engineering Constructs. *Journal of Cognitive Engineering and Decision Making*, 2(2):140–160, June 2008.

[8] A. Blidaru, S. L. Smith, and D. Kulić. Assessing user specifications for robot task planning. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 72–79, Aug 2018.

[9] Nils Wilde, Alexandru Blidaru, Stephen L. Smith, and Dana Kulić. Improving User Specifications for Robot Behavior through Active Preference Learning: Framework and Evaluation. *arXiv:1907.10412 [cs]*, July 2019.

[10] Sebastian Thrun. Toward a framework for human-robot interaction. *Hum.-Comput. Interact.*, 19(1):9–24, June 2004.

[11] Jean Scholtz. Theory and evaluation of human robot interactions. In *36th Annual Hawaii International Conference on System Sciences*, January 2003.

[12] Michael A. Goodrich and Alan C. Schultz. Humanrobot interaction: A survey. *Foundations and Trends in Human Computer Interaction*, 1(3):203–275, 2008.

[13] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common Metrics for Human-robot Interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, HRI '06, pages 33–40, New York, NY, USA, 2006. ACM.

[14] Guide: Guide to Human Performance Measurements (AIAA G-035A-2000). In Life Sciences and Systems Committee, editor, *Guide: Guide to Human Performance Measurements (AIAA G-035A-2000)*. American Institute of Aeronautics and Astronautics, Inc., Washington, DC, January 2000.

[15] M. R. Endsley. Situation awareness global assessment technique (SAGAT). In *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*, pages 789–795 vol.3, May 1988.

[16] J. Scholtz, B. Antonishek, and J. Young. Evaluation of a human-robot interface: Development of a situational awareness methodology. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of The*, pages 10 pp.–, January 2004.

[17] Raja Parasuraman, Scott Galster, Peter Squire, Hiroshi Furukawa, and Christopher Miller. A flexible delegation-type interface enhances system performance in human supervision of multiple robots: Empirical studies with RoboFlag. *IEEE Transactions on systems, man, and cybernetics-part A: Systems and Humans*, 35(4):481–493, 2005.

[18] Mica R. Endsley and Esin O. Kiris. The Out-of-the-Loop Performance Problem and Level of Control in Automation. *Human Factors*, 37(2):381–394, June 1995.

[19] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In Peter A. Hancock and Najmedin Meshkati, editors, *Advances in Psychology*, volume 52 of *Human Mental Workload*, pages 139–183. North-Holland, January 1988.

[20] Neville A Stanton, Paul M Salmon, Laura A Rafferty, Guy H Walker, Chris Baber, and Daniel P Jenkins. *Human Factors Methods: A Practical Guide for Engineering and Design*. CRC Press, 2017.

[21] Cynthia Breazeal and Brian Scassellati. A context-dependent attention system for a social robot. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1146–1151, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[22] C. Breazeal. Social interactions in HRI: The robot view. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):181–186, May 2004.

[23] J. W. Crandall, M. A. Goodrich, D. R. Olsen, and C. W. Nielsen. Validating human-robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):438–449, July 2005.

[24] A. Lampe and R. Chatila. Performance measure for the evaluation of mobile robot autonomy. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 4057–4062, May 2006.

[25] J. W. Crandall, M. A. Goodrich, D. R. Olsen, and C. W. Nielsen. Validating human-robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):438–449, July 2005.

[26] Jacob W Crandall. *Towards Developing Effective Human-Robot Systems*. PhD thesis, Brigham Young University. Department of Computer Science, 2003.

[27] Jacob W Crandall and Michael A Goodrich. Measuring the intelligence of a robot and its interface. In *Proc. of PERMIS*, 2003.

[28] J. W. Crandall and M. A. Goodrich. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1290–1295 vol.2, 2002.

[29] G. Yang and G. T. Anderson. An experimental study of environmental complexity as seen by robots. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3102–3106, October 2011.

[30] G. T. Anderson and G. Yang. A proposed measure of environmental complexity for robotic applications. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 2461–2466, October 2007.

[31] S. H. Young, T. A. Mazzuchi, and S. Sarkani. A Framework for Predicting Future System Performance in Autonomous Unmanned Ground Vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1192–1206, July 2017.

[32] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics. *The International Journal of Robotics Research*, 32(11):1238–1274, 2015.

[33] W. D. Smart and L. Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, volume 4, pages 3404–3410 vol.4, May 2002.

[34] Peter Stone, Richard S Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[35] W Bradley Knox, Peter Stone, and Cynthia Breazeal. Training a robot via human feedback: A case study. In *International Conference on Social Robotics*, pages 460–470. Springer, 2013.

[36] Aude G Billard, Sylvain Calinon, and Rüdiger Dillmann. *Learning from humans*. Springer, 2016.

[37] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009.

[38] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):33, 2008.

[39] Yun Lin, Shaogang Ren, Matthew Clevenger, and Yu Sun. Learning grasping force from demonstration. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1526–1531. IEEE, 2012.

[40] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 391–398. ACM, 2012.

[41] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[42] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[43] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*, pages 1133–1141, 2012.

[44] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. pages 4299–4307, 2017.

[45] Amit Bhatia, Lydia E Kavraki, and Moshe Y Vardi. Sampling-based motion planning with temporal goals. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2689–2696. IEEE, 2010.

[46] Shashank Srinivas, Ramtin Kermani, Kangjin Kim, Yoshihiro Kobayashi, and Georgios Fainekos. A graphical language for ltl motion and mission planning. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 704–709. IEEE, 2013.

[47] Cameron Finucane, Gangyuan Jing, and Hadas Kress-Gazit. Ltlmop: Experimenting with language, temporal logic and robot control. In *IROS 2010*, pages 1988–1993. IEEE, 2010.

[48] M. Lahijanian and M. Kwiatkowska. Specification revision for markov decision processes with optimal trade-off. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7411–7418, Dec 2016.

[49] Jesper Karlsson, Cristian-Ioan Vasile, Jana Tumova, Sertac Karaman, and Daniela Rus. Multi-vehicle motion planning for social optimal mobility-on-demand. In *2018*

IEEE International Conference on Robotics and Automation (ICRA), pages 7298–7305. IEEE, 2018.

[50] K Hauser. The Minimum Constraint Removal Problem with Three Robotics Applications. *Int'l. J. of Robotics Research*, 33(1):5–17, 2014.

[51] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9. Berlin, Germany, 2013.

[52] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3804–3809. IEEE, 2011.

[53] Nils Wilde, Dana Kulic, and Stephen L. Smith. Learning User Preferences in Robot Motion Planning through Interaction. In *IEEE International Conference on Robotics and Automation*, 2018.

[54] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active Reward Learning. *Robotics: Science and Systems (RSS)*, 10, 2014.

[55] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

[56] Thane Somers and Geoffrey A. Hollinger. Human–robot planning and learning for marine data collection. *Autonomous Robots*, 40(7):1123–1137, Oct 2016.

[57] L Adrián León, Ana C Tenorio, and Eduardo F Morales. Human interaction for effective reinforcement learning. In *European Conf. Mach. Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2013)*, volume 3, 2013.

[58] J. E. Laird, K. Gluck, J. Anderson, K. D. Forbus, O. C. Jenkins, C. Lebiere, D. Salvucci, M. Scheutz, A. Thomaz, G. Trafton, R. E. Wray, S. Mohan, and J. R. Kirk. Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017.

[59] Riad Akrour, Marc Schoenauer, and Michèle Sebag. APRIL: Active preference learning-based reinforcement learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7524 LNAI(PART 2):116–131, 2012.

[60] Samantha Krening and Karen M Feigh. Interaction algorithm effect on human experience with reinforcement learning. *ACM Transactions on Human-Robot Interaction (THRI)*, 7(2):16, 2018.

[61] Shengbo Guo and Scott Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 289–296, 2010.

[62] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pages 766–774, 2010.

[63] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. Learning preferences for manipulation tasks from online coactive feedback. *International Journal of Robotics Research*, 34(10):1296–1313, 2015.

[64] Chandrayee Basu, Mukesh Singhal, and Anca D. Dragan. Learning from richer human guidance: Augmenting comparison-based learning with feature queries. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, pages 132–140, New York, NY, USA, 2018. ACM.

[65] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[66] Holly A. Yanco and Jill L. Drury. Rescuing interfaces: A multi-year study of human-robot interaction at the AAAI Robot Rescue Competition. *Autonomous Robots*, 22(4):333–352, May 2007.

[67] Holly A. Yanco, Adam Norton, Willard Ober, David Shane, Anna Skinner, and Jack Vice. Analysis of Human-robot Interaction at the DARPA Robotics Challenge Trials. *Journal of Field Robotics*, 32(3):420–444, May 2015.

[68] Adam Norton, Willard Ober, Lisa Baraniecki, Eric McCann, Jean Scholtz, David Shane, Anna Skinner, Robert Watson, and Holly Yanco. Analysis of human–robot interaction at the DARPA Robotics Challenge Finals. *The International Journal of Robotics Research*, 36(5-7):483–513, June 2017.

[69] DJI Support. DJI Terra - Mapping and 2D Reconstruction [Tutorial]. https://youtu.be/AZH-hR_GM0c.

[70] Aerial Drones. http://us.yuneec.com.

[71] How Industrial Drones Work. https://www.kespry.com/how-it-works/.

[72] Aeryon Labs Inc. AeryonLive - Viewer Designated Marker (VDM). https://youtu.be/Yv-xtL_wfiU.

[73] OTTO Motors. Agv vs. sdv: A comparison of automated material transport white paper, November 2016. https://www.ottomotors.com/agv-vs-sdv [Online; posted 07-November-2016].

[74] G. Yang and G. T. Anderson. An experimental study of environmental complexity as seen by robots. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3102–3106, October 2011.

[75] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. Active Preference-Based Learning of Reward Functions. In *Robotics: Science and Systems (RSS)*, July 2017.

[76] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Inc., 4th edition, 2007.

[77] Kevin G Jamieson and Robert Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems*, pages 2240–2248, 2011.

[78] Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the system usability scale. *International Journal of HumanComputer Interaction*, 24(6):574–594, 2008.

[79] Y. Cui and S. Niekum. Active reward learning from critiques. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6907–6914, May 2018.